KIẾN TRÚC MÁY TÍNH
(Computer Architecture)

- **Khoa Kỹ thuật máy tính**
- GV: TS. **Vũ Đức Lung**
- Email: lungvd@uit.edu.vn

- Thời gian:
  - Lý Thuyết: 45 tiết (3 TC)
- Điểm số:
  - Điểm thi giữa HK: 30%
  - Điểm thi cuối kỳ: 70%

Mục đích môn học

Nhằm trang bị cho sinh viên các kiến thức cơ bản nhất về kiến trúc một máy tính.

- Lịch sử
- Chức năng và nguyên lý hoạt động của các bộ phận
- Cách biểu diễn dữ liệu, tính toán trong máy tính
- Cách chế tạo, thiết kế các mạch Logic số cơ bản
- Các kiến trúc bộ lệnh trong các loại máy tính CISC và RICS
- Các nguyên lý hoạt động của bộ xử lý

Nội dung

Chương 1 : Giới thiệu

Chương 2 : Các bộ phận cơ bản của máy tính

Chương 3 : Biểu diễn dữ liệu

Chương 4 : Mạch Logic số

Chương 5 : Mạch tuần tự

Chương 6 : Kiến trúc bộ lệnh

Chương 7 : Tổ Chức bộ xử lý

Chương 8 : Hệ Thống bộ nhớ

Tài liệu học tập & tham khảo

1. Vũ Đức Lung. Giáo trình kiến trúc máy tính. Trường ĐH Công nghệ thông tin, ĐHQG TP.HCM, 2009.

2. Cấu trúc máy tính cơ bản, tổng hợp và biên dịch VN-Guide, nhà xuất bản thống kê, 2005.

3. Võ Văn Chín, Nguyễn Hồng Vân, Phạm Hữu Tài. Giáo trình kiến trúc máy tính. ĐH Cần Thơ, 2005.

4. M. Abd-El-Barr, H. El-Rewini, Fundamentals of Computer Organization and Architecture, Wiley, 2005

5. Patterson, D. A., and J. L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*, 3rd ed. San Mateo, CA: Morgan Kaufman, 2004

Slides + bài tập:

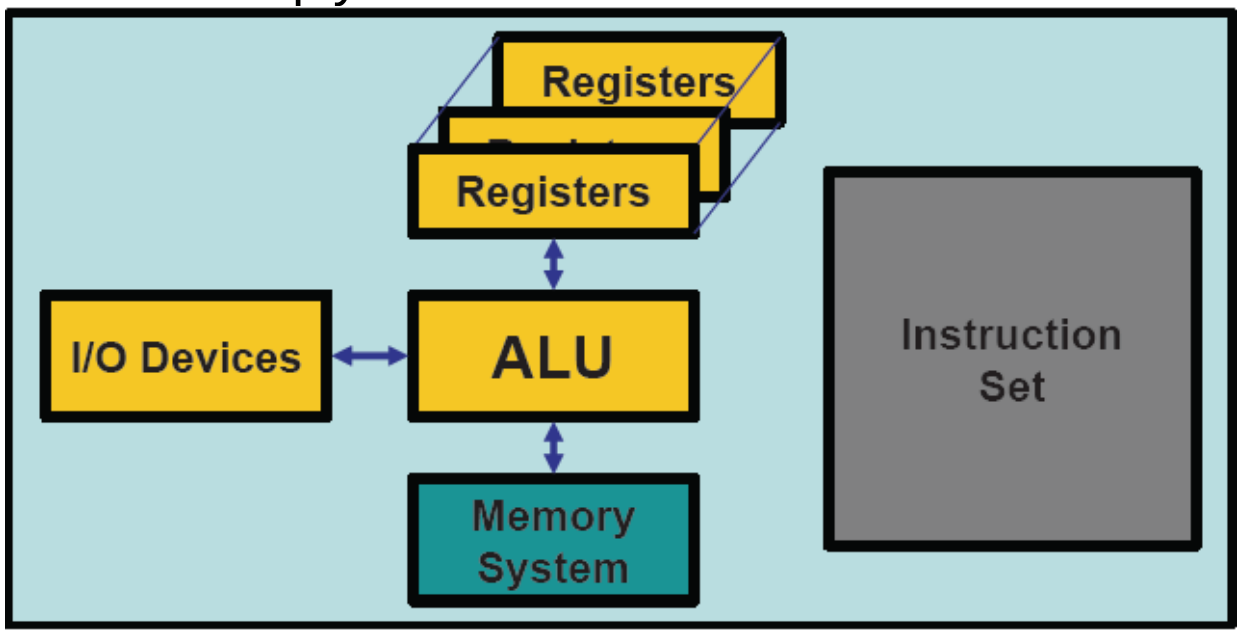
http://groups.google.com/group/ca-vdlung?hl=en

Mục đích - nắm bắt cơ bản về:

1. Một số khái niệm cơ bản về kiến trúc máy tính
2. Lịch sử phát triển của máy tính qua các thế hệ máy tính:
    1. ***http://www.computersciencelab.com***
    2. ***http://www.computerhistory.org***
3. Khuynh hướng hiện tại cho phát triển ngành máy tính
4. Phân loại máy tính
5. Các dòng CPU Intel

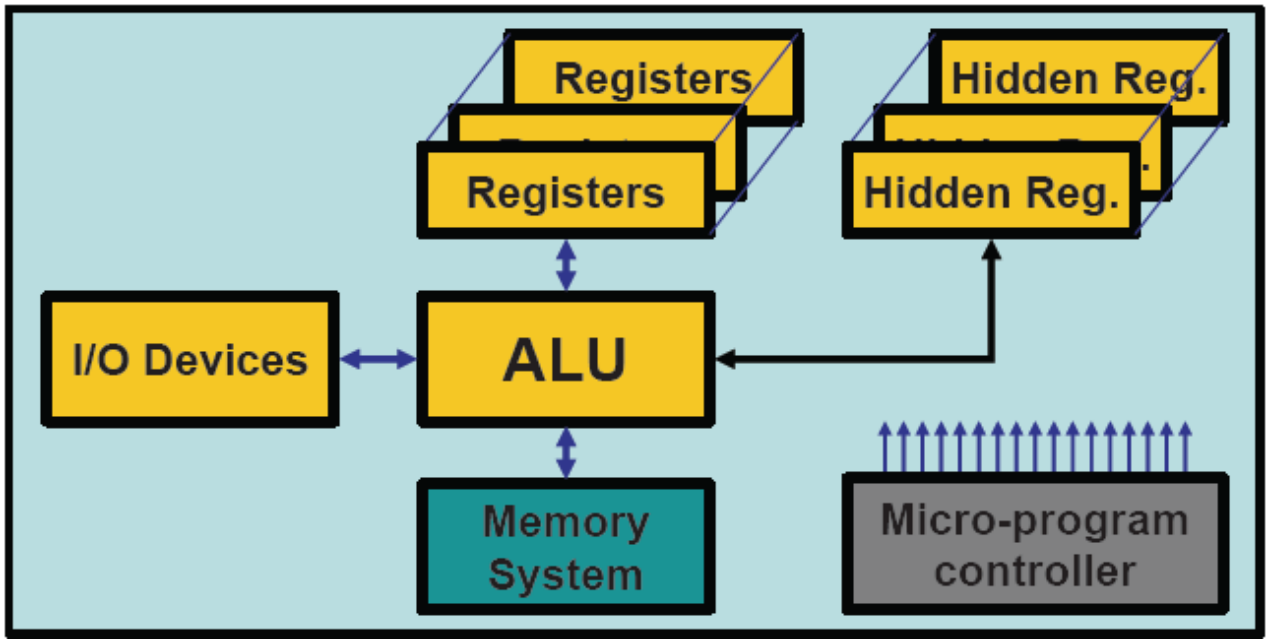
Kiến trúc máy tính

- Kiến trúc máy tính đề cập đến những thuộc tính hệ thống mà lập trình viên có thể quan sát được.
- Đó là các thuộc tính có ảnh hưởng trực tiếp đến việc thực thi một chương trình,
- *Ví dụ*: Tập chỉ thị của máy tính, số bit được sử dụng để biểu diễn dữ liệu, cơ chế nhập/xuất, kỹ thuật định địa chỉ bộ nhớ, v.v...
- e.g. Is there a multiply instruction?



Tổ chức máy tính

- Tổ chức máy tính quan tâm đến các đơn vị vận hành và sự kết nối giữa chúng nhằm hiện thực hóa những đặc tả về kiến trúc,
- *Ví dụ*: tín hiệu điều khiển, giao diện giữa máy tính với các thiết bị ngoại vi, kỹ thuật bộ nhớ được sử dụng
- e.g. Is there a hardware multiply unit or is it done by repeated addition?



Học Kiến trúc máy tính để làm gì?

- To be a professional in any field of computing today, you should not regard the computer as just a black box that executes programs by magic.
- You should understand a computer system's functional components, their characteristics, their performance, and their interactions.
- You need to understand computer architecture in order to build a program so that it runs more efficiently on a machine.
- When selecting a system to use, you should be able to understand the tradeoff among various components, such as CPU clock speed vs. memory size.

IEEE/ACM Computer Curricula

Máy tính là gì?

Là máy xử lý dữ liệu, thực thi tự động dưới sự điều khiển của một danh sách các câu lệnh lưu trong bộ nhớ

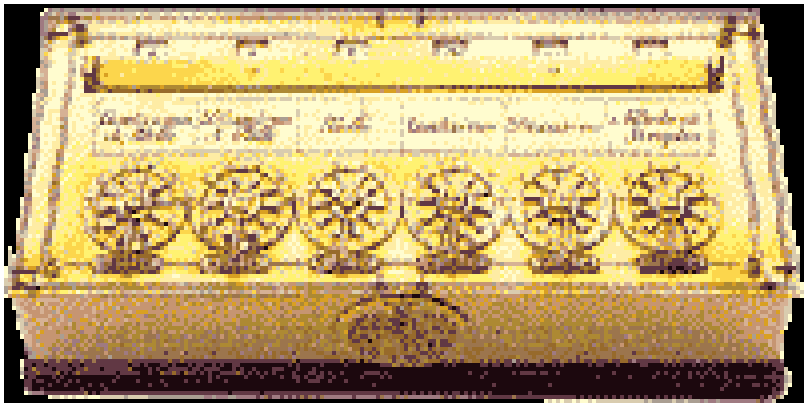
Thị phần bộ vi xử lý

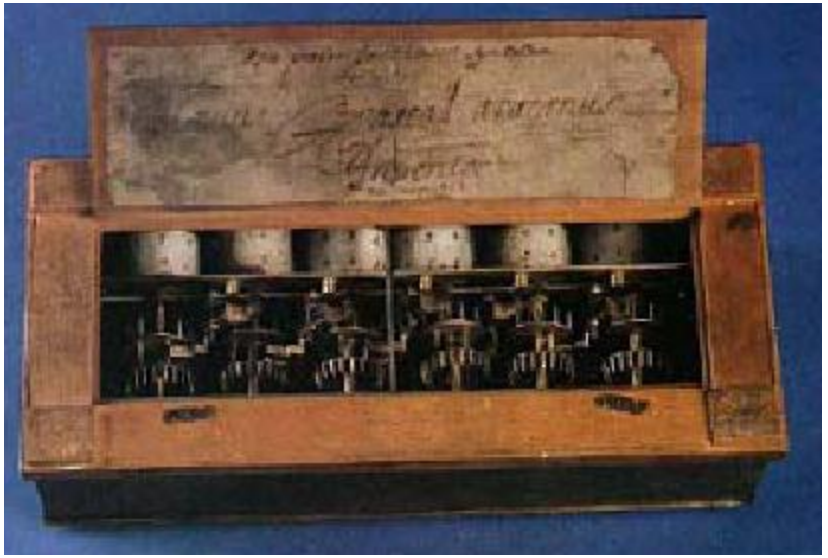
2. Lịch sử phát triển máy tính

- Thế hệ zero –máy tính cơ học (1642-1945)

Thế hệ zero –máy tính cơ học (1642-1945)

Năm 1642 Pascal phát minh ra máy tính đầu tiên với 2 phép tính + và -



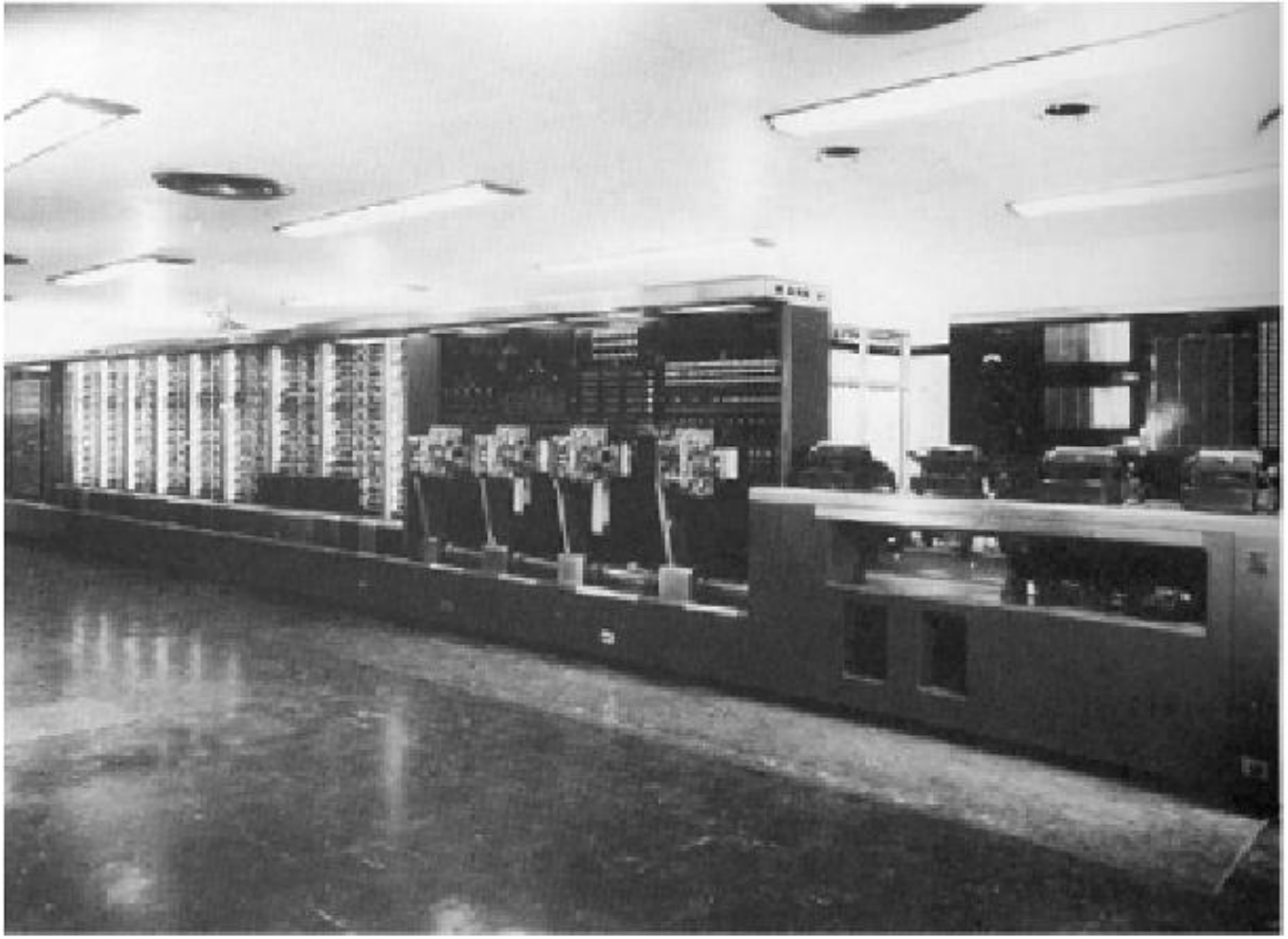


Thế hệ zero –máy tính cơ học (1642-1945)

Năm 1672 Gotfrid vilgelm ***Leibnits*** chế tạo ra máy tính với 4 phép tính cơ bản (+-*/)

- 1834 Babbage (Anh) – máy tính có 4 bộ phận: bộ nhớ, bộ tính toán, thiết bị nhập, thiết bị xuất
- 1936 K. Zuse (Đức) máy trên cơ sở rơle (relay)
- 1944 G. Iken (Mỹ) – Mark I
  - nặng 5 tấn,
  - cao 2.4 m,
  - dài 15 m,
  - chứa 800 km dây điện

Thế hệ zero –máy tính cơ học (1642-1945)



Havard Mark I

Thế hệ I – bóng đèn điện (1945-1955)

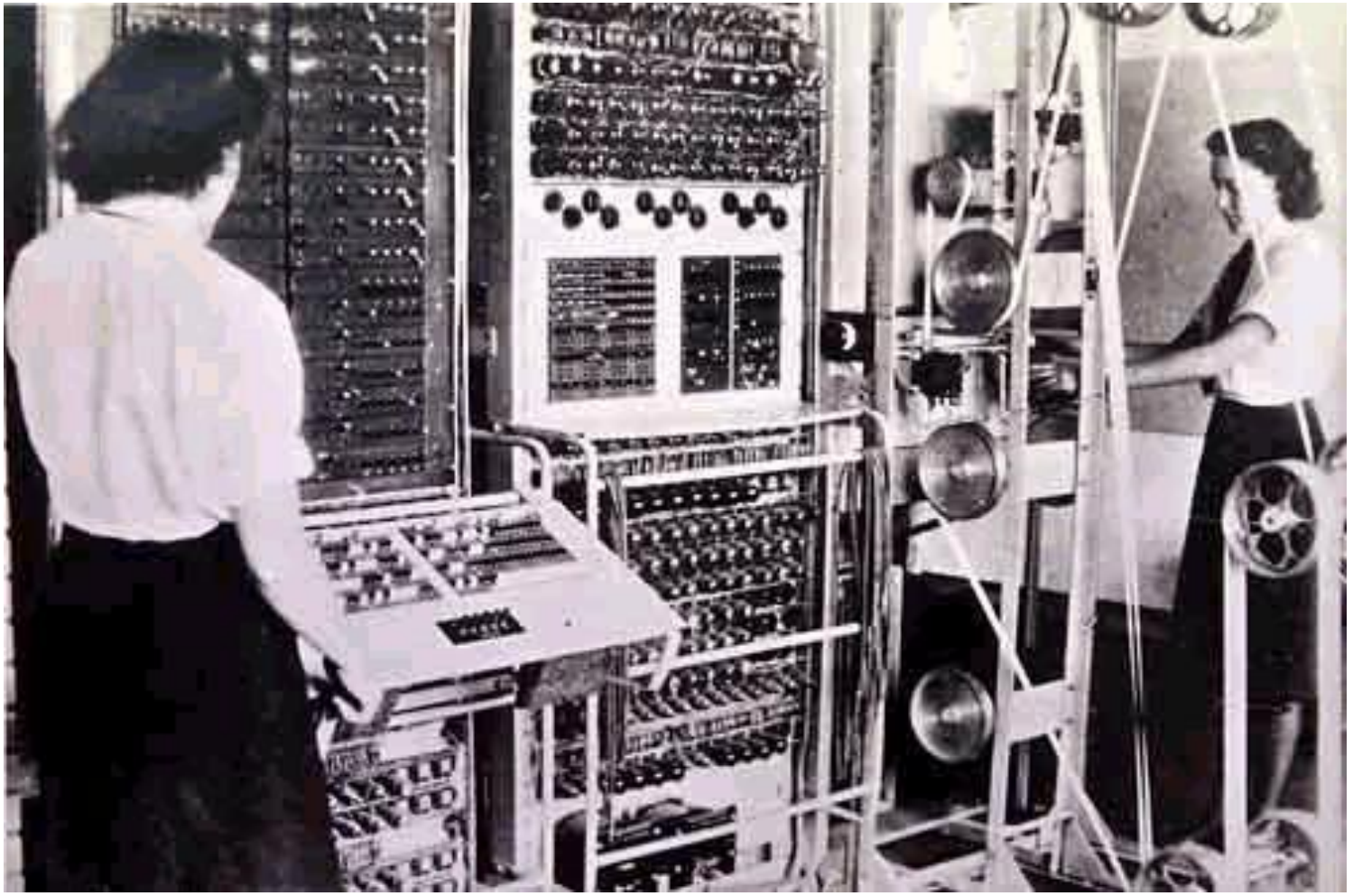
1. 1943 máy tính COLOSSUS (Anh)

- 2000 bóng đèn chân không
- Giữ bí mật suốt 30 năm

Alain Turing

Bóng đèn chân không

Thế hệ I – bóng đèn điện (1945-1955)



COLOSSUS

Thế hệ I – bóng đèn điện (1945-1955)

2. Máy tính ENIAC 1943 (Mỹ)

Dự án chế tạo máy **ENIAC** (Electronic Numerical Integrator and Computer) được BRL (Ballistics Research Laboratory – Phòng nghiên cứu đạn đạo quân đội Mỹ) bắt đầu vào năm 1943 dùng cho việc tính toán chính xác và nhanh chóng các bảng số liệu đạn đạo cho từng loại vũ khí mới.

❑ *Các thông số:*

- ✓ 18000 bóng đèn chân không.
- ✓ Nặng hơn 30 tấn.
- ✓ Tiêu thụ một lượng điện năng vào khoảng 140kW và chiếm một diện tích xấp xỉ 1393 m^2.
- ✓ 5000 phép cộng /s.
- ✓ Đặc biệt sử dụng hệ đếm thập phân.

Thế hệ I – bóng đèn điện (1945-1955)

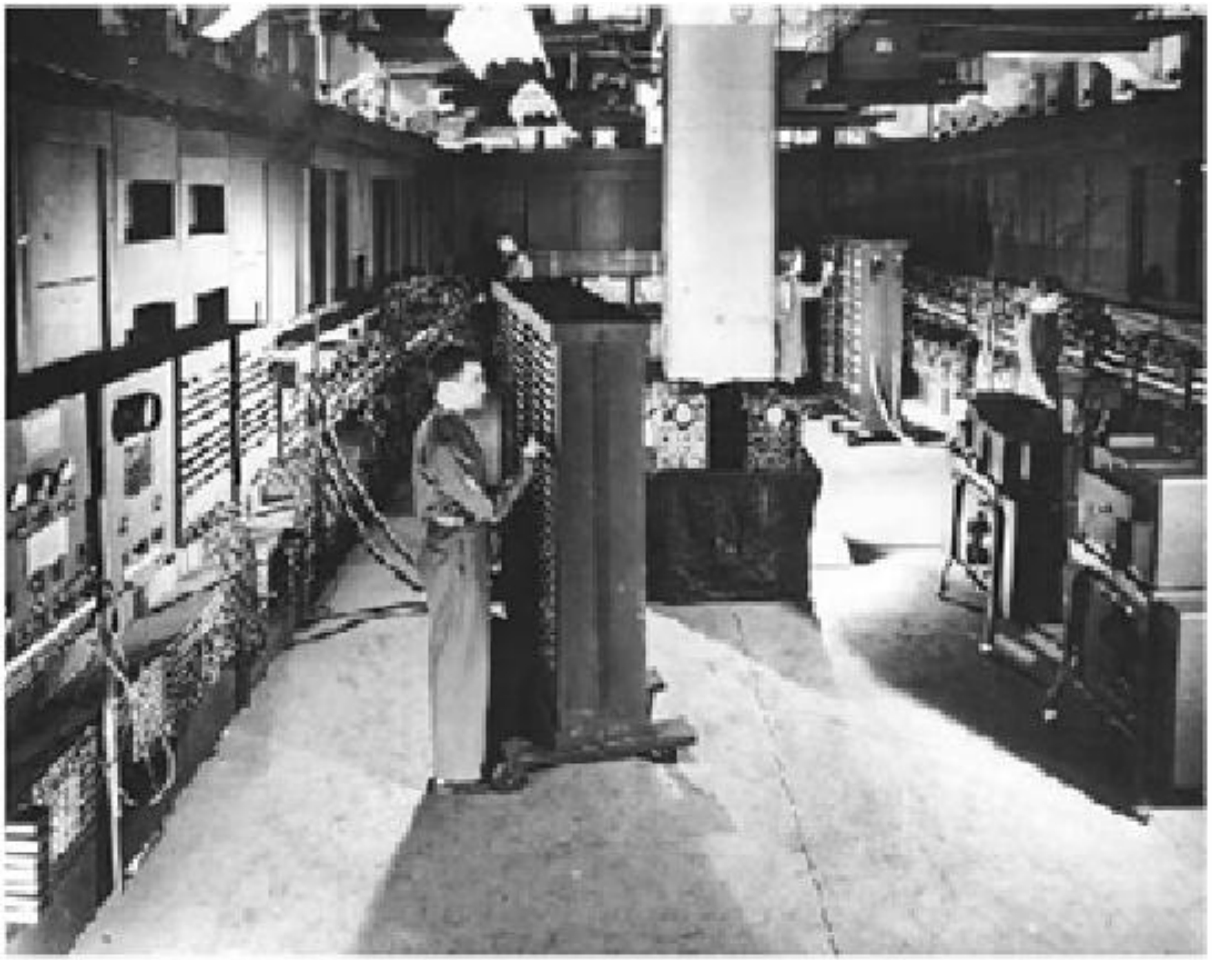
Bộ nhớ của ENIAC

- 20 "bộ tích lũy".
- Mỗi bộ có khả năng lưu giữ một số thập phân có 10 chữ số.
- Mỗi chữ số được thể hiện bằng một vòng gồm 10 đèn chân không.

Điểm khác biệt giữa ENIAC & các máy tính khác:
ENIAC sử dụng hệ đếm thập phân chứ không phải nhị phân như ở tất cả các máy tính khác.

- Máy ENIAC bắt đầu hoạt động vào tháng 11/1945.

Thế hệ I – bóng đèn điện (1945-1955)



Máy ENIAC

Thế hệ I – bóng đèn điện (1945-1955)

3. Máy tính Von Neumann 1952 (Mỹ)

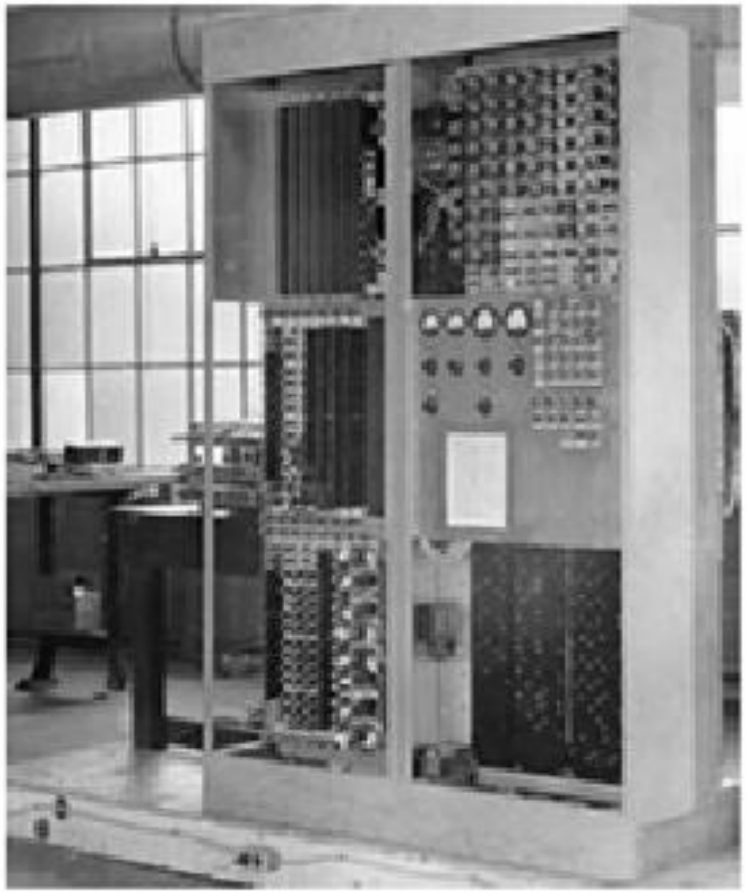
Nhà toán học John ***von Neumann***(Hungary), một cố vấn của dự án ENIAC, đưa ra 1945, trong một bản đề xuất về một loại máy tính mới có tên gọi ***EDVAC*** (Electronic Discrete Variable Computer).

- ***2500 bóng đèn điện tử.***

- Chương trình lưu trong bộ nhớ (Không cần phải nối dây lại như máy ENIAC).

Thế hệ I – bóng đèn điện (1945-1955)

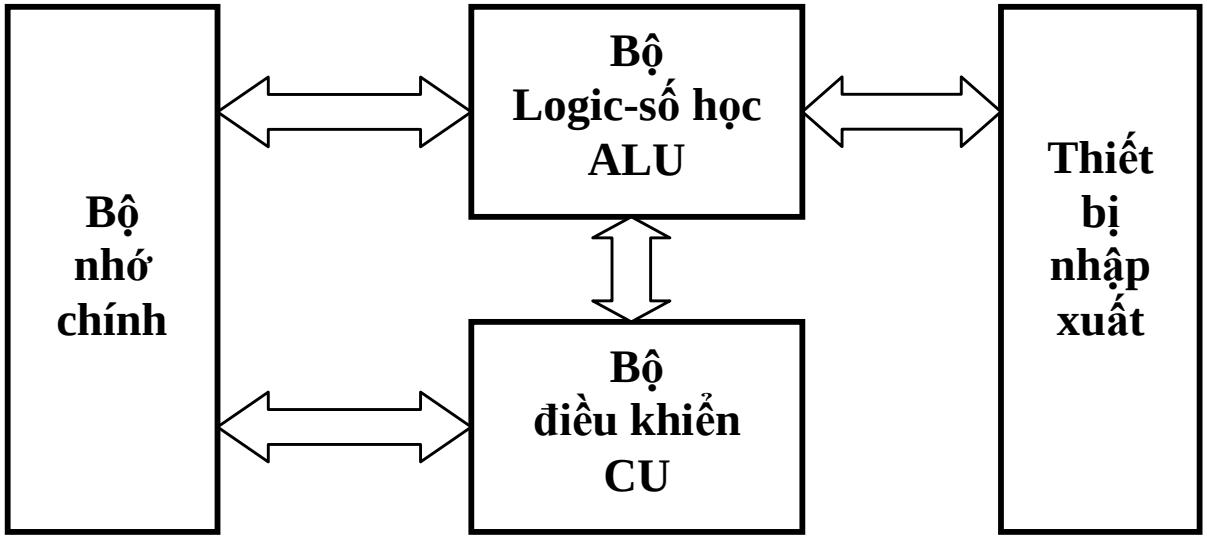
John ***von Neumann***



Máy EDVAC

Thế hệ I – bóng đèn điện (1945-1955)

1952 ra đời ***IAS*** (Institute for Advanced Studies) tại học viện nghiên cứu cao cấp Princeton, Mỹ.



Cấu trúc của máy IAS

1952 máy tính Von Neumann ra đời – ***cơ sở cho kiến trúc máy tính hiện đại (bit 1,0).***

Đặc tính của IAS

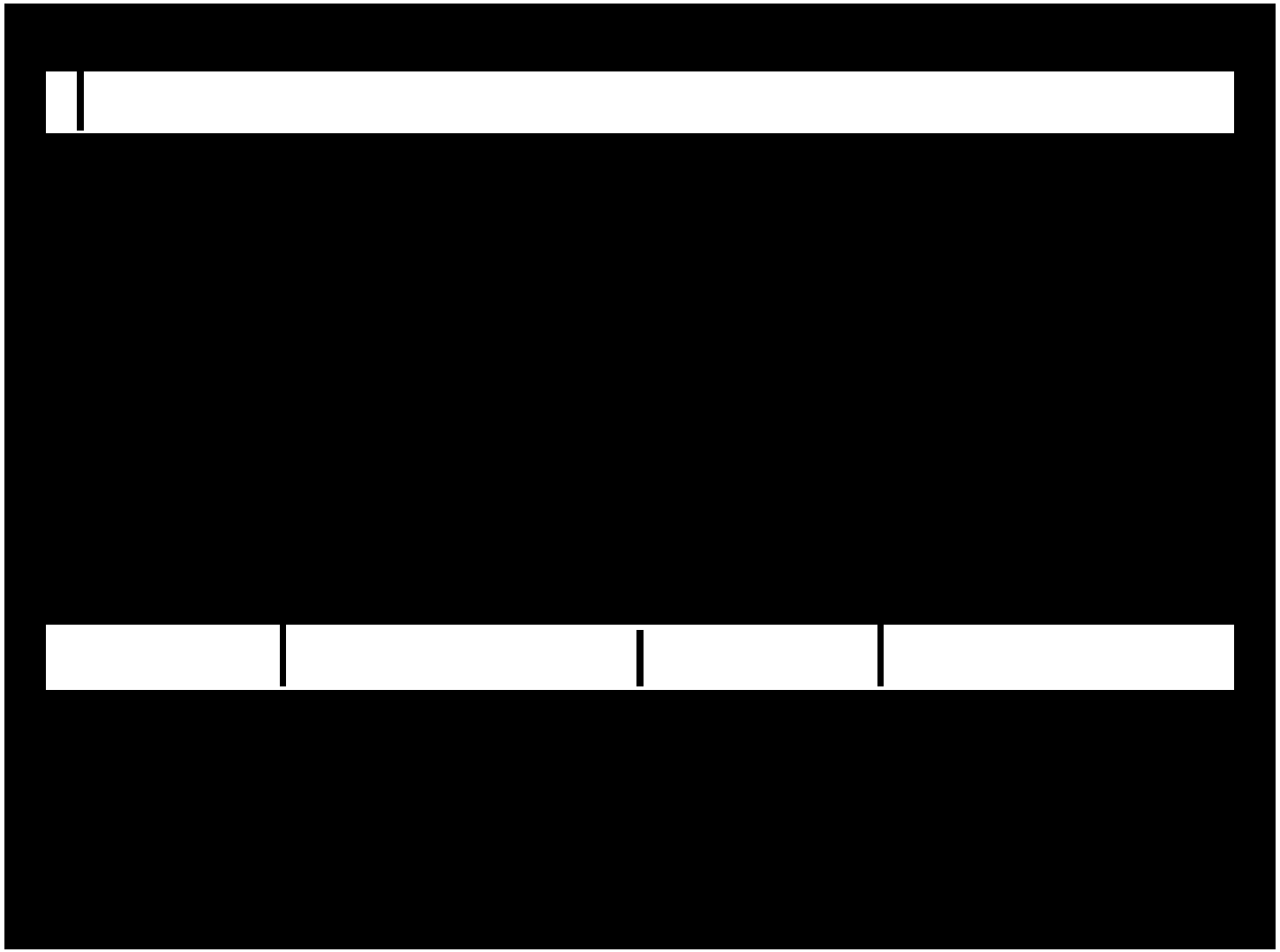
Kỹ thuật stored-program

- Chương trình được đưa vào bộ nhớ chính đã được địa chỉ hóa.
- Máy tính dùng bộ đếm chương trình để thi hành tuần tự các lệnh.

Bộ nhớ

- 1000 vị trí lưu trữ, gọi là word.
- 1 word = 40 bit.
- Mỗi số được biểu diễn bằng 1 bit dấu và một giá trị 39 bit.
- 1 word có thể chứa 2 chỉ thị 20 bit, với mỗi chỉ thị gồm:
  + Mã thao tác 8 bit (Op code) đặc tả thao tác sẽ được thực hiện.
  + Địa chỉ 12 bit (Address) định hướng đến một word trong bộ nhớ (địa chỉ này đi từ 0 đến 999).

Các dạng thức bộ nhớ của máy IAS



Thế hệ II – transistor (1955-1965)

- Sự thay đổi đầu tiên trong lĩnh vực máy tính điện tử xuất hiện khi có sự thay thế đèn chân không bằng ***đèn bán dẫn***.
- Đèn bán dẫn nhỏ hơn, rẻ hơn, tỏa nhiệt ít hơn trong khi vẫn có thể được sử dụng theo cùng cách thức của đèn chân không để tạo nên máy tính

Năm 1947 - ***Bardeen***, ***Brattain*** và ***Shockley*** của phòng thí nghiệm ***Bell Labs*** đã phát minh ra ***transistor*** và đã được giải Nobel vật lý năm 1956.

Thế hệ II – transistor (1955-1965)

Máy IBM 7030 chứa 150,000 transistor (1959)

Thế hệ II – transistor (1955-1965)

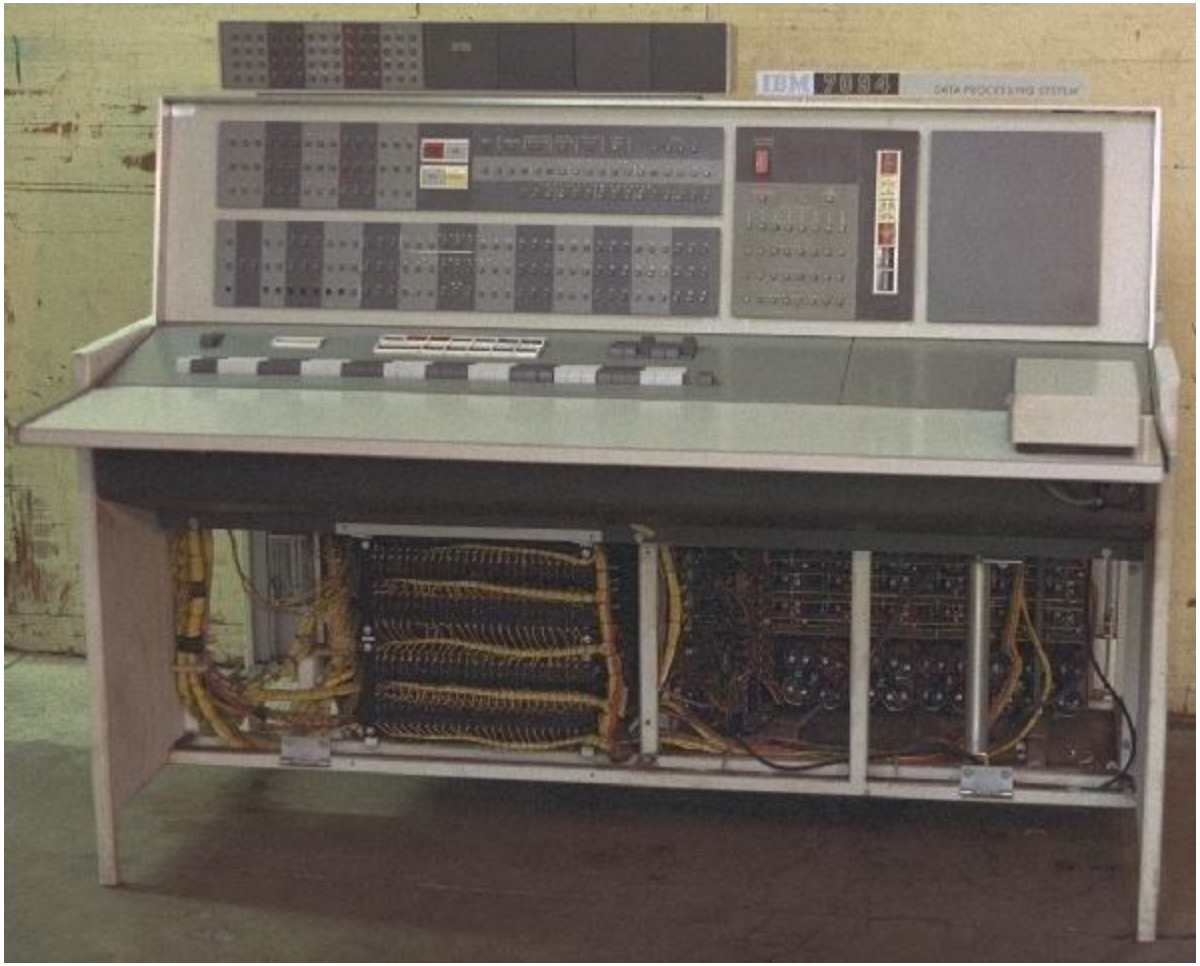
Trong thế hệ này nổi tiếng nhất là 2 máy:

- ***PDP-1*** của DEC là máy tính nhỏ gọn nhất thời bấy giờ. DEC (Digital Equipment Corporation) được thành lập vào năm 1957 và cũng trong năm đó cho ra đời sản phẩm đầu tiên của mình là PDP-1.
  - ✓4 K word (1 word= 18 bit).
  - ✓Chu kỳ 5 ms.
  - ✓Giá 120,000$.
- ***IBM 7094***.
  - ✓32 K word (1 word = 16 bit).
  - ✓Chu kỳ 2 ms.
  - ✓Giá 1,000,000$.

Thế hệ II – transistor (1955-1965)

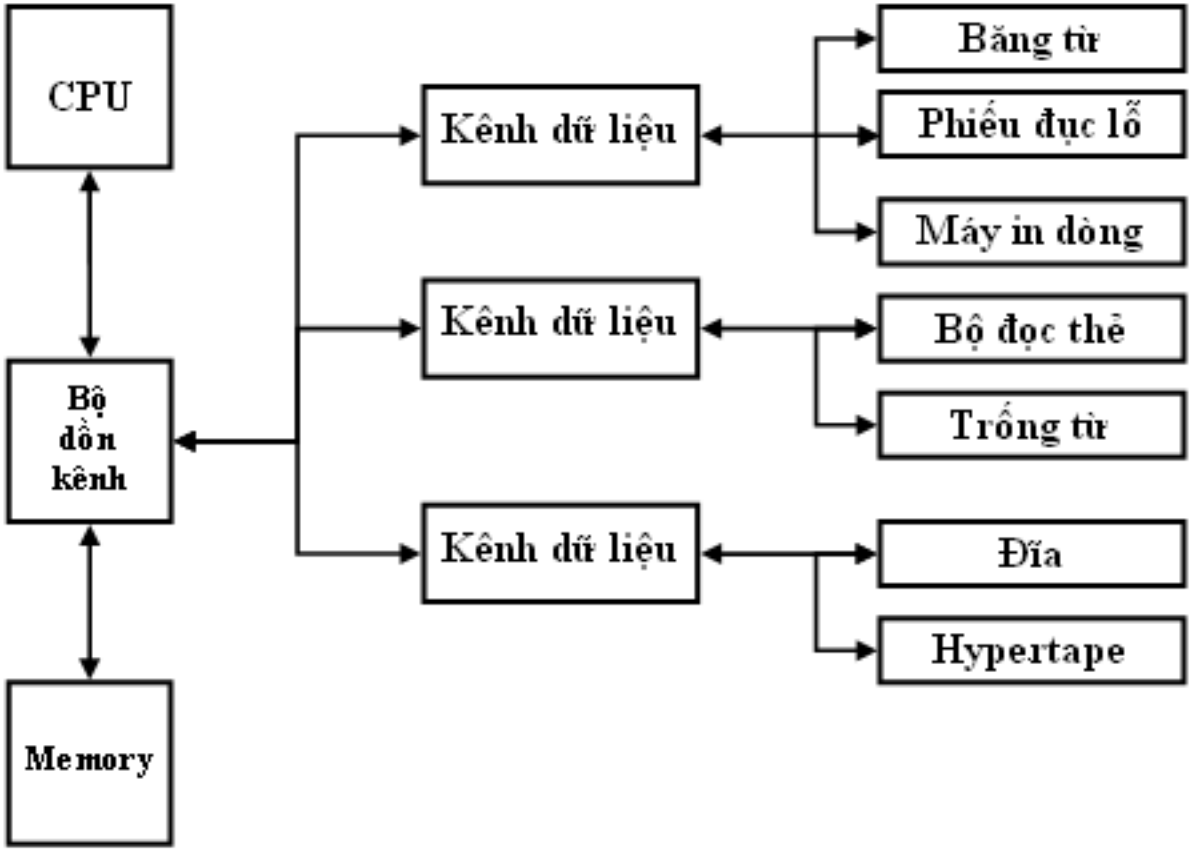
Máy IBM 7094

Thế hệ II – transistor (1955-1965)



An IBM 7094 console

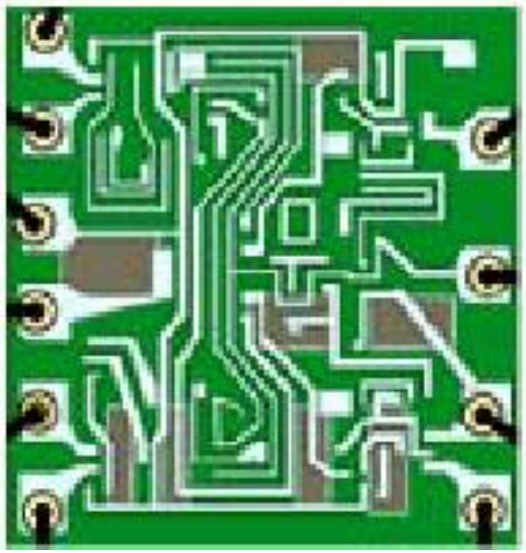
Thế hệ II – transistor (1955-1965)



Cấu trúc máy IBM 7094

Thế hệ III – mạch tích hợp (1965-1980)

1958 ***Jack Kilby*** và ***Robert Noyce*** đã cho ra đời một công nghệ mới, công nghệ mạch tích hợp (***Integrated circuit – IC***)



Thế hệ III – mạch tích hợp (1965-1980)

- Máy ***IBM System 360*** được IBM đưa ra vào năm 1964 là họ máy tính công nghiệp đầu tiên được sản xuất một cách có kế hoạch.

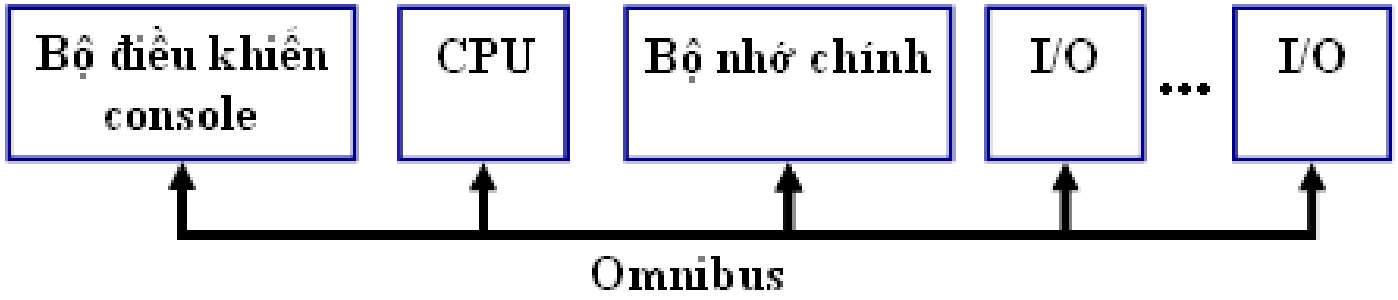
- Đặc biệt khái niệm họ máy tính bao gồm các máy tính tương thích nhau là một khái niệm mới và hết sức thành công. Nhờ đó mà một chương trình được viết cho máy này cũng sẽ dùng được trên những máy khác cùng họ với nó.

Khái niệm này đã được dùng cho đến ngày nay.

Thế hệ III – mạch tích hợp (1965-1980)

Máy DEC PDP-8

- PDP-8 đã sử dụng một cấu trúc rất phổ dụng hiện nay cho các máy mini và vi tính: cấu trúc đường truyền.
- Đường truyền PDP-8, được gọi là Omnibus, gồm 96 đường tín hiệu riêng biệt, được sử dụng để mang chuyển tín hiệu điều khiển, địa chỉ và dữ liệu.

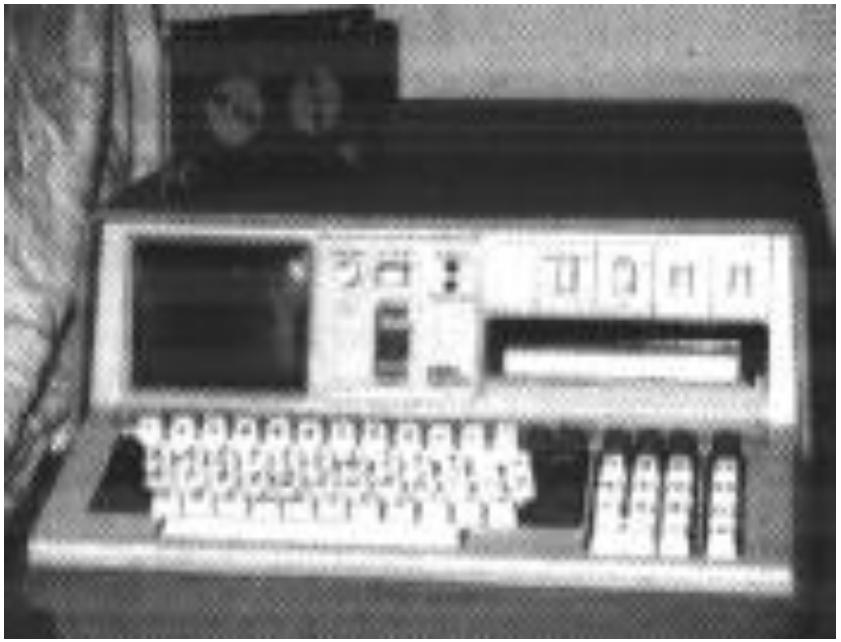


Cấu trúc PDP-8

Thế hệ III – mạch tích hợp (1965-1980)

1975 máy tính cá nhân đầu tiên (Portable computer) IBM 5100 ra đời, tuy nhiên máy tính này đã không gặt hái được thành công nào.

- Băng từ.
- Nặng 23 Kg.
- 10,000$.
- Khả năng lập trình trên Basic.
- Màn hình 16 dòng, 64 ký tự.
- Bộ nhớ <=64Kbyte.



-1979 chương trình Sendmail ra đời bởi 1 sinh viên ĐHTH California, Berkely university cho ra đời BSD UNIX (Berkely Software Distribution)

Thế hệ IV – máy tính cá nhân (1980-?)

- Sự xuất hiện của công nghệ ***VLSI (very large scale integrated)*** cho phép trên một bản mạch có thể sắp sếp hàng triệu transistor.

- Từ đây bắt đầu kỷ nguyên của máy tính cá nhân.

Thế hệ IV – máy tính cá nhân (1980-?)

- 1981 ra đời máy IBM PC trên cơ sở CPU Intel 8088 và dùng hệ điều hành MS-DOS của Microsoft.

- 1983 PC/XT (Extended Technology) với HDD 10 MB hoặc 20 MB với giá chỉ có 1,995$.

Khối các nước XHCN

-1950 tại trường cơ khí chính xác và quang học (CNTT bây giờ): máy tính toán điện cỡ lớn đầu tiên ra đời với mục đích giải quyết các bài toán khoa học và kỹ thuật phức tạp.

- 1953 tại đại học toán, viện hàn lâm – máy Strela.

- 1954 PC – Ural 1-16 Minsk, Kiev…

- Nói chung trong thời kỳ đầu tuy có ra sau một thời gian, nhưng hầu hết đều có máy tính tương đương với xu hướng của thế giới.

EC-1840

Khối các nước XHCN

- EC-1840,41- tương đương với 8086
- EC-1842,1843 - tương đương 80286
- Từ 1849 trở đi các máy tính dùng CPU của Intel.

PC	Năm bắt đầu SX	Năm kết thúc SX	Số lượng
EC-1840	1986	1989	7461
EC-1841	1987	1995	83937
EC-1842	1988	1996	10193
EC-1843	1990	1993	3012
EC-1849	1990	1997	4966
EC-1851	1991	1997	3142
EC-1863	1991	1997	3069
BM2001	1994	-	1074

3. Khuynh hướng hiện tại và tương lai

- Tăng tần số xung đồng hồ.

- Xử lý song song.

- Đa lõi CPU.

- Máy tính thông minh, trí tuệ nhân tạo: LISP và PROLOG.

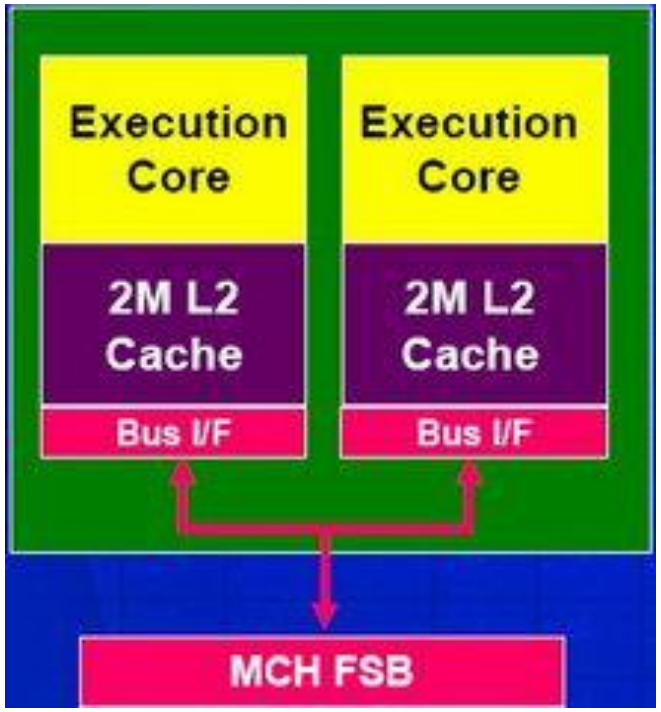
- ***ASIMO*** (*Advanced Step Innovative Mobility)*.

3. Khuynh hướng hiện tại và tương lai

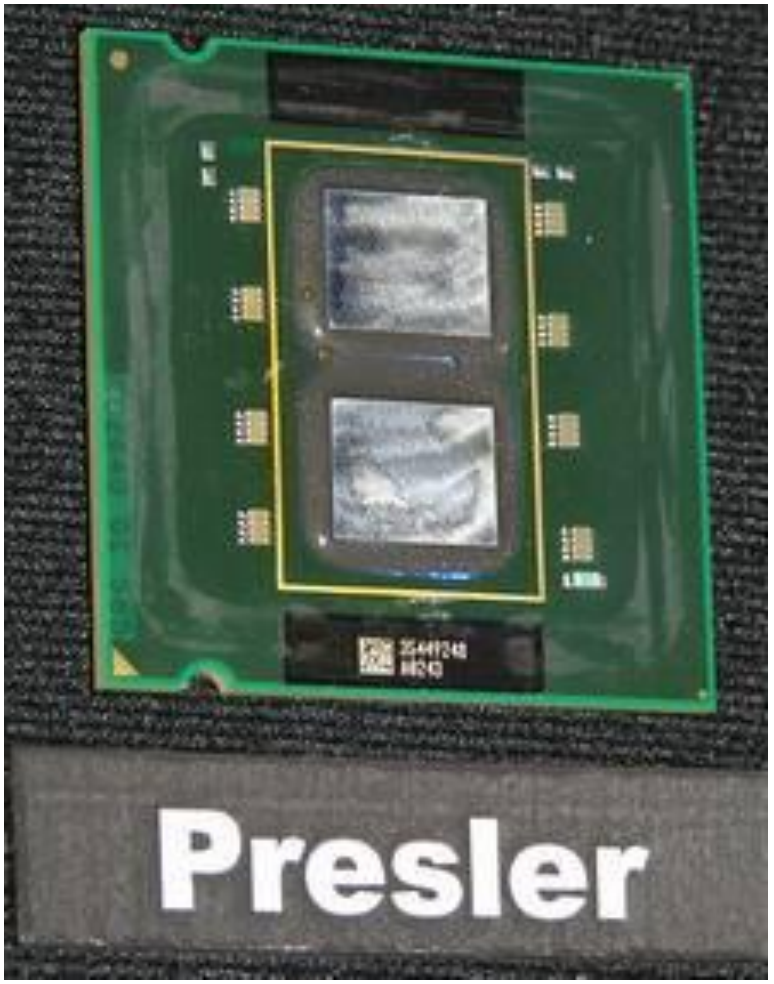
Các bộ sử lý đa lõi

- 1999 CPU 2 lõi kép đầu tiên ra đời (IBM Power4 cho máy chủ).
- 2001 bắt đầu bán ra thị trường Power4.
- 2002 AMD và Intel cùng thông báo về việc thành lập CPU đa lõi của mình.
- 2004 CPU lõi kép của Sun ra đời UltraSPARS IV
- 2005 Power5.
- 03/2005 CPU Intel lõi kép x86 ra đời, AMD – Opteron, Athlon 64X2.
- 20-25/05/2005 AMD bắt đầu bán Opteron 2xx,
- 26/05 Intel Pentium D, 31/05 AMD – bán Athlon 64X2.

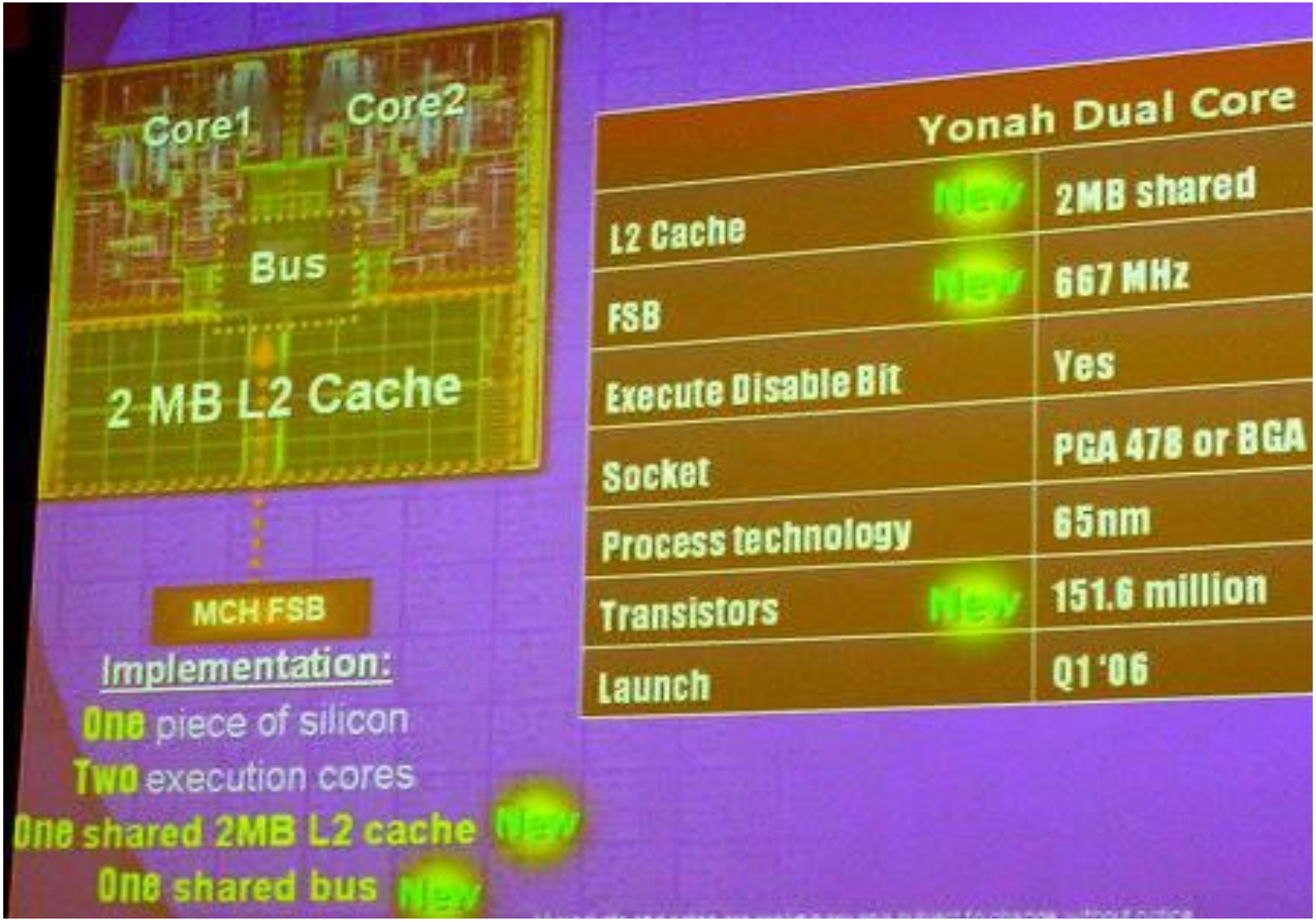
3. Khuynh hướng hiện tại và tương lai



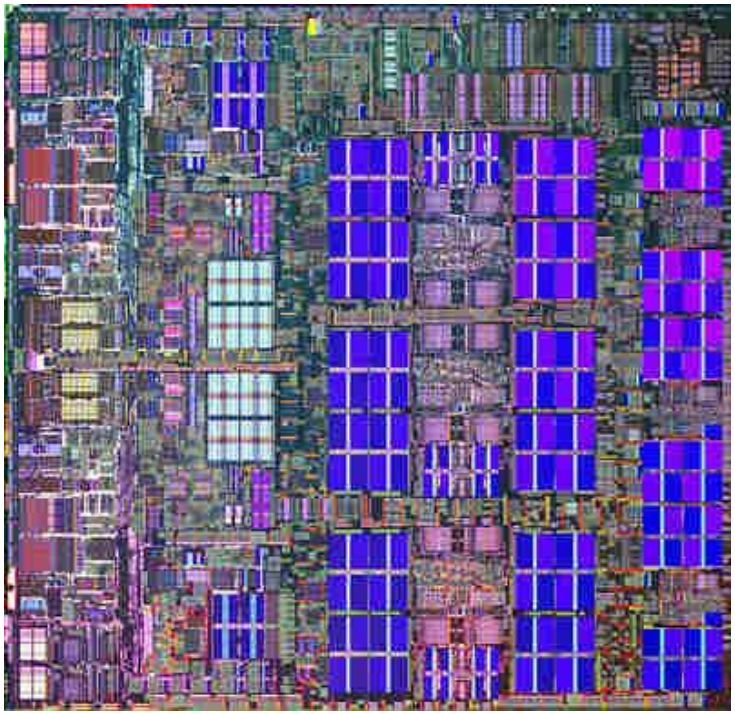
Presler 65nm

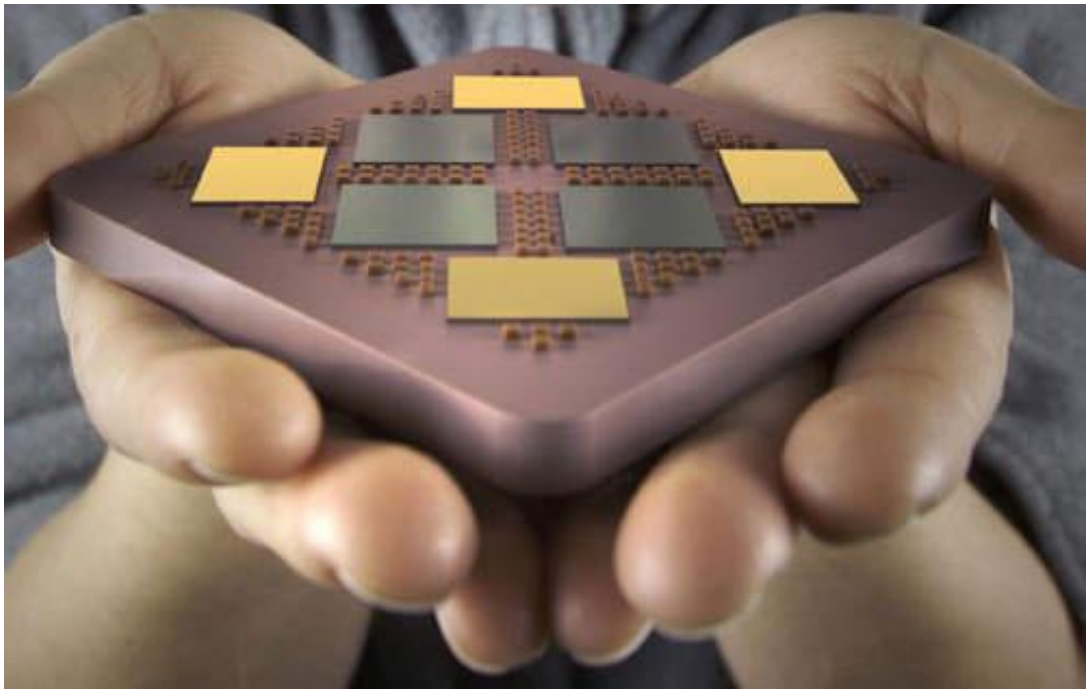


Yonah Dual Core



CPU Power5





3. Khuynh hướng hiện tại và tương lai

Một trong những siêu máy tính hàng đầu của thế giới (8192 CPU, 7,3 Tfops)

26/06/2007: supercomputer Blue Gene/L với 128 dãy, 130 ngàn CPU, 360 Tfops, 267 triệu USD. (***MDGRAPE-3*: 1PFlops)**

Moscow State University Supercomputer

- 350 teraflops

4. Phân loại máy tính

a) Các siêu máy tính (Super Computer):

>1 triệu USD, IBM Deep Blue, Blue Gene, ***MDGRAPE-3***

b) Các máy tính lớn (Mainframe)

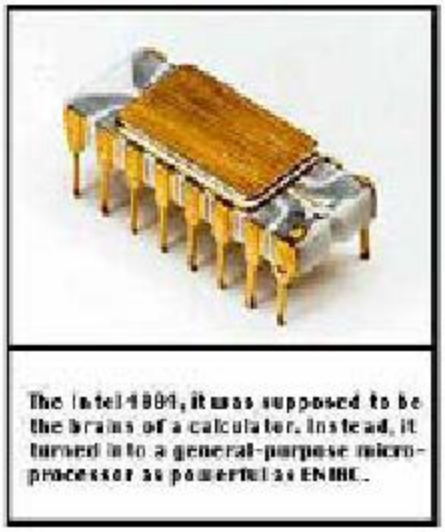
từ vài trăm – 1 triệu USD. IBM mainframes, Unisys ClearPath mainframes, Hitachi zSeries (z800), Hewlett-Packard, Fujitsu BS2000 and Fujitsu-ICL VME

c) Máy tính mini (Minicomputer)

Vài chục đến vài trăm ngàn USD.
Control Data's CDC 160A and CDC 1700,DEC PDP and VAX series, Hewlett-Packard HP 3000 series, HP 2100 series, HP1000 series. IBM midrange computers,Texas Instruments TI-990

d) Máy vi tính (Microcomputer or personal computer)

5. Các dòng Intel



- 1970 bộ CPU 4004 (4 bit) của Intel

trên 1 chip đầu tiên ra đời.

- 1972 CPU Intel 8008 (8 bit).

- 1974 CPU 8080, 1978 CPU 8086 (16 bit).

- 1979 CPU 8088 (8 bit).

- 1981 máy tính IBM PC đầu tiên ra đời trên cơ sở CPU Intel 8088 và hệ điều hành MS DOS.

- 1982 CPU 80286 (16 bit).

- 1985 CPU 80386 (32 bit), 89-486, 93-Pentium…

5. Các dòng CPU Intel

Tên gọi	Năm giới thiệu	Số lượng Transistors	Microns	Tốc độ đồng hồ	Data width	MIPS
8080	1974	6,000	6	2 MHz	8 bits	0.64
8088	1979	29,000	3	5 MHz	16 bits 8-bit bus	0.33
80286	1982	134,000	1.5	6 MHz	16 bits	1
80386	1985	275,000	1.5	16 MHz	32 bits	5
80486	1989	1,200,000	1	25 MHz	32 bits	20
Pentium	1993	3,100,000	0.8	60 MHz	32 bits 64-bit bus	100
Pentium II	1997	7,500,000	0.35	233 MHz	32 bits 64-bit bus	~300
Pentium III	1999	9,500,000	0.25	450 MHz	32 bits 64-bit bus	~510
Pentium 4	2000	42,000,000	0.18	1.5 GHz	32 bits 64-bit bus	~1,700
Pentium 4 "Prescott"	2004	125,000,000	0.09	3.6 GHz	32 bits 64-bit bus	~7,000

5. Các dòng CPU Intel

Pentium D,
Core 2 Duo,
Intel® Core™2 Quad processor ,
Intel® Core™2 Extreme processor
Intel® Core™ i7 processor
Intel® Core™ i7 processor Extreme Edition

Corei7
- 45nm
- 4 nhân
- Turbo Boost
- Intel® Quickpath
- Cache L3 8MB
- Intel® Desktop Board DX58SO Extreme Series
- chipset Intel® X58 Express
- Gia 330$ (6tr3)

Câu hỏi và bài tập

1. Nắm bắt các khái niệm cơ bản
2. Lịch sử phát triển của máy tính
3. Các xu hướng
4. Phân loại máy tính

Chương II
Các bộ phận cơ bản của máy tính

Nội dung

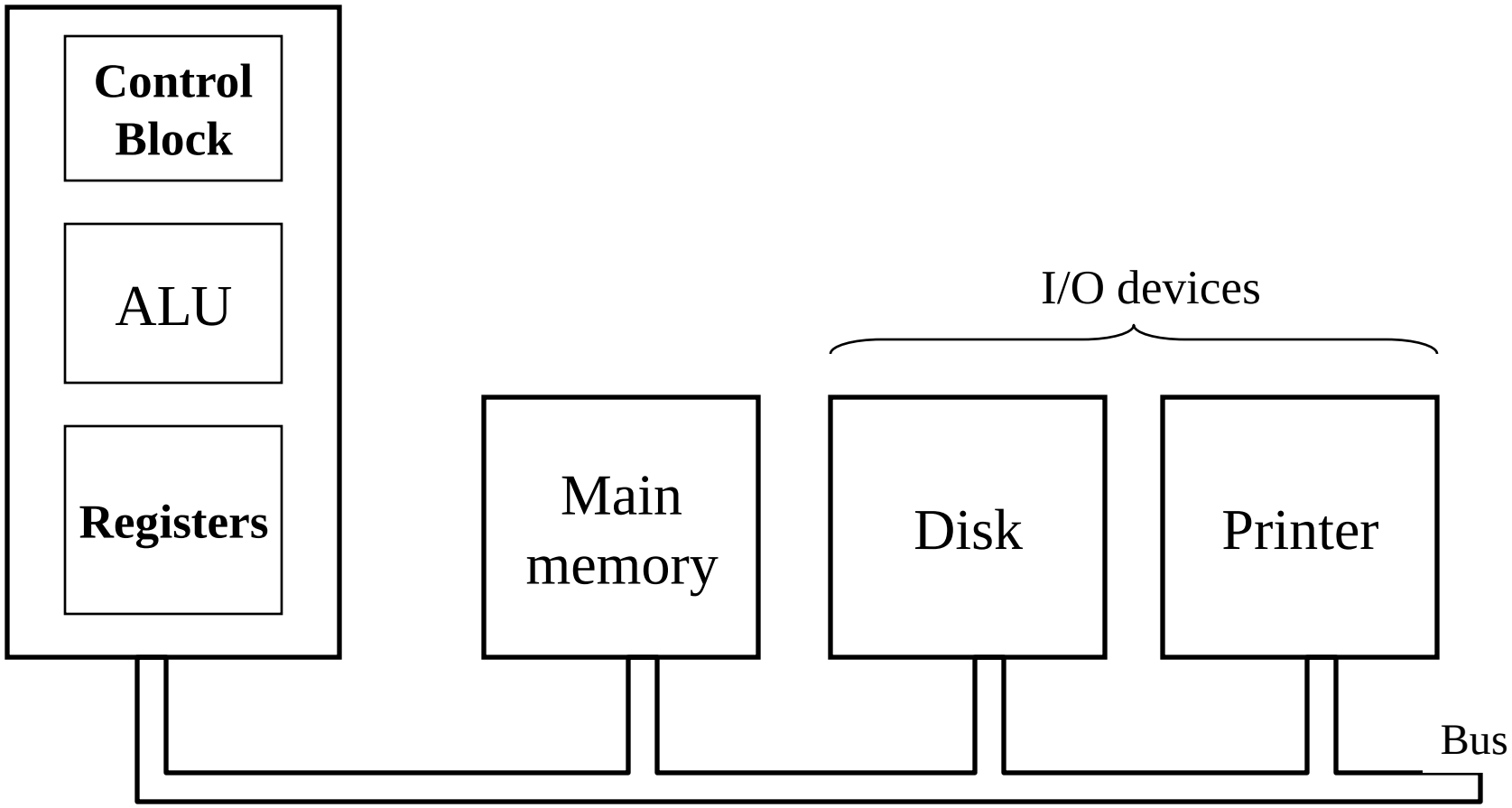
1. Bộ xử lý (CPU)
2. Bản mạch chính (Mainboard)
3. Ổ đĩa mềm (FDD)
4. Ổ đĩa cứng (HDD)
5. Ổ CD và DVD
6. Bộ nhớ RAM và ROM
7. Bàn phím (Keyboard)
8. Chuột (Mouse)
9. Card màn hình (VGA Card)
10. Màn hình (Monitor)
11. Card mạng (Network adapter) và Modem

1. Bộ vi xử lý (CPU)

- Bộ vi xử lý CPU (*Central Processing Unit)* là cốt lõi của một máy vi tính.
- CPU 8 bit, 16 bit, 32 bit, 64 bit.
- Công ty sản xuất CPU – Intel, AMD, Cyrix, IBM, HP…

1. Bộ vi xử lý (CPU)

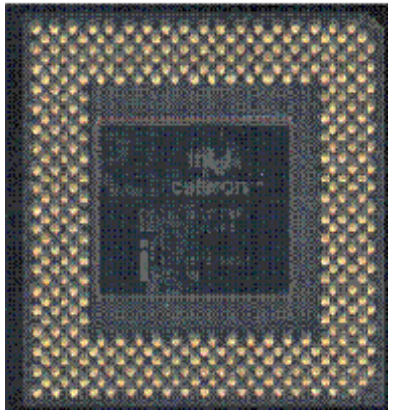
Central Processing Unit - CPU



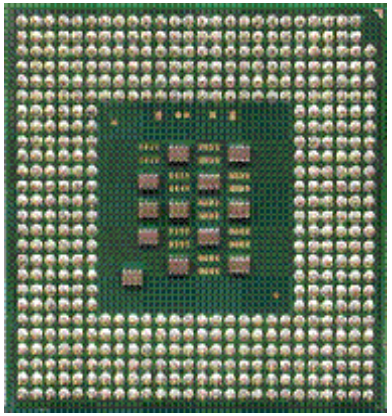
Tổ chức máy tính theo hướng BUS đơn giản

1. Bộ vi xử lý (CPU)

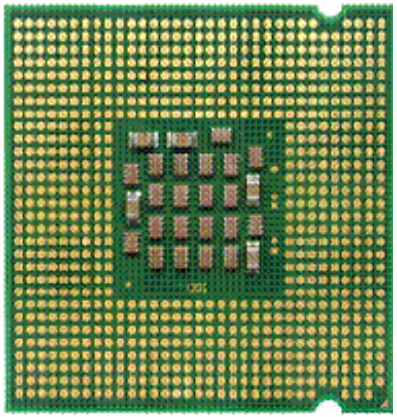
- Các thông số chính
  - ***Hãng sản xuất và model (Processor make and model)***
  - ***Dạng Socket (Socket type):***



Socket 370



Socket 478



Socket 775

  - ***Tốc độ đồng hồ xung (Clock Speed - CS)***
  - ***Tốc độ đường truyền chủ (Host-bus speed, front-side bus (FSB))***
  - ***Kính thước bộ nhớ đệm (Cache size)***

Các loại Socket

Socket	Khả năng nâng cấp	CPU gốc	CPU có thể nâng cấp
478	có	Celeron, Celeron D, Pentium 4	Celeron D, Pentium 4
754	tốt	Sempron, Athlon 64	Sempron, Athlon 64
775	rất tốt	Celeron D, Pentium 4, DuoCore, Core2Duo	Celeron D, Pentium 4, Pentium D
1156	rất tốt	Core i3, Core i5, Core i7	
1366	rất tốt	Core i7 Extreme	
939	rất tốt	Athlon 64, Athlon 64/FX	Athlon 64, Athlon 64/FX, Athlon 64 X2
940	rất tốt	Athlon 64 FX, Opteron	Athlon 64 FX, Opteron
AM2, AM3	rất tốt	Athlon II, Phenom	

1. Bộ vi xử lý (CPU)

AMD ATHLON 64 x2 - 5200+ (2.6)	Socket **AM2**	2x1152KB	Bus 2000	**140/ 73**
AMD ATHLON 64 x2 - 5600+ (2.8)	Socket **AM2**	2x1152KB	Bus 2000	**162/ 97**
AMD ATHLON 64 x2 - 6000+ (3.0)	Socket **AM2**	2x1152KB	Bus 2000	**183/ 101**

INTEL P 4 925 – 3.0D GHz.	Socket **775**	**2x2MBK**	Bus 800	**77/69**
INTEL Duo Core-E2180(2.0GHz)	Socket **775**	**1M Duo Core**	Bus 800	**100/62**
INTEL Core2 Duo-E4500(2.2GHz)	Socket **775**	**2M Core 2 Duo**	Bus 800	**151**
INTEL Core2 Duo-E6320(1.86GHz)	Socket **775**	**4M Core 2 Duo**	Bus 1066	**182**
INTEL Core2 Duo-E6420(2.13GHz)	Socket **775**	**4M Core 2 Duo**	Bus 1066	**202**

1. Bộ vi xử lý (CPU)

<u>*VÍ DỤ*</u>: P4 2.8Ghz (511)/Socket 775/ Bus 533/ 1024K/ Prescott CPU

- P4 - CPU Pentium 4, 2.8 Ghz - tốc độ xung đồng hồ của vi xử lý, 511 - chất lượng và vị thế của con CPU trong toàn bộ các sản phẩm thuộc cùng dòng.
- Socket 775, chỉ loại khe cắm của CPU.
- Bus 533, chỉ tốc độ "lõi" của đường giao tiếp giữa CPU và mainboard.
- 1024K, chỉ bộ nhớ đệm của vi xử lý. Đây là vùng chứa thông tin trước khi đưa vào cho vi xử lý trung tâm (CPU) thao tác.
- Prescott chính là tên một dòng vi xử lý của Intel. Dòng vi xử lý này có khả năng xử lý video siêu việt nhất trong các dòng vi xử lý cùng công nghệ của Intel. Tuy nhiên, đây là dòng CPU tương đối nóng, tốc độ xung đồng hồ tối đa đạt 3.8 Ghz.

1. Bộ vi xử lý (CPU)

❑ *Sự khác biệt cơ bản giữa AMD và Intel*

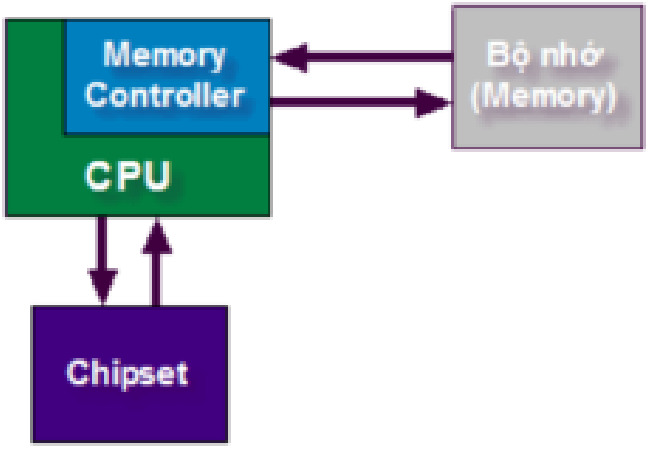
a) Cách đặt tên

- ***AMD***: Athlon 64 X2 4800+, Athlon X2 BE-2350

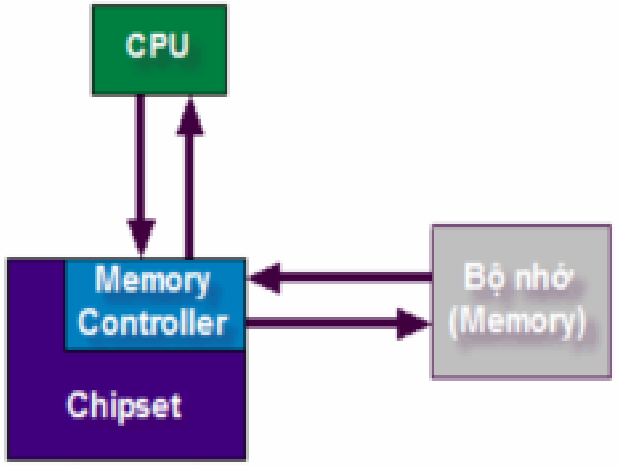
[2 ký tự biểu thị nhóm]-[ký tự biểu thị dòng][3 ký tự biểu thị model]

- ***Intel*** : Pentium 4 3GHz , Pentium 4 630, Core2 Duo-E4500

b) Các công nghệ tiêu biểu



Bố trí memory kiểu AMD



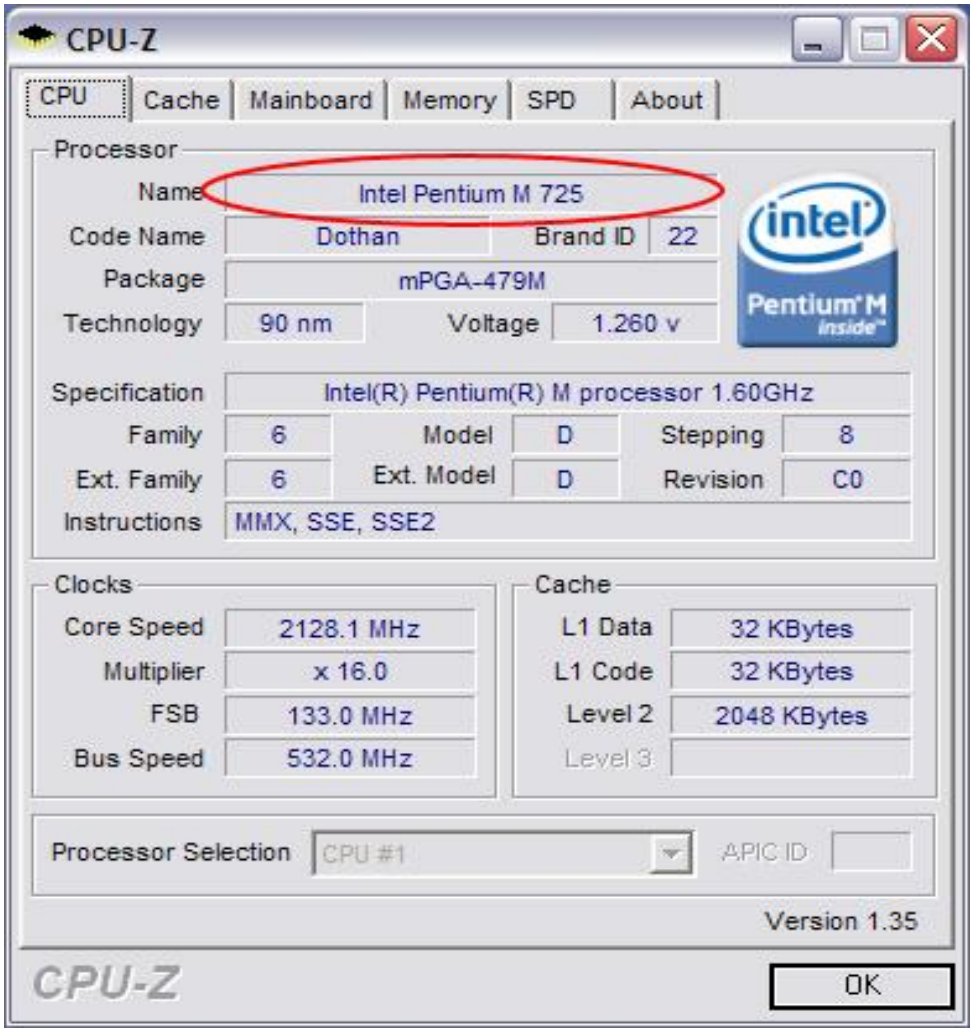
Bố trí memory kiểu Intel

1. Bộ vi xử lý (CPU)

❑ ***Sự khác biệt cơ bản giữa AMD và Intel***

- Memory Controller
- HyperTransport
- Hyper Threading

c) <u>Tỏa nhiệt</u>

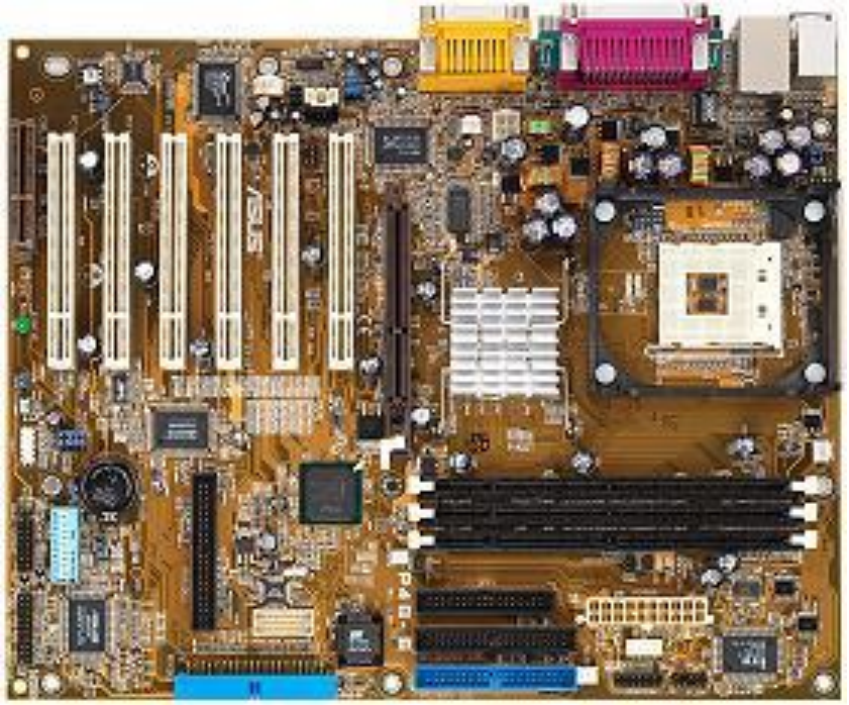


2. Bản mạch chính (mainboard)

-Bản mạch chính chứa đựng những linh kiện điện tử và những chi tiết quan trọng nhất của một máy tính cá nhân như:

- ➢Bộ vi xử lý CPU (central processing unit),
- ➢Hệ thống bus
- ➢Các vi mạch hỗ trợ.

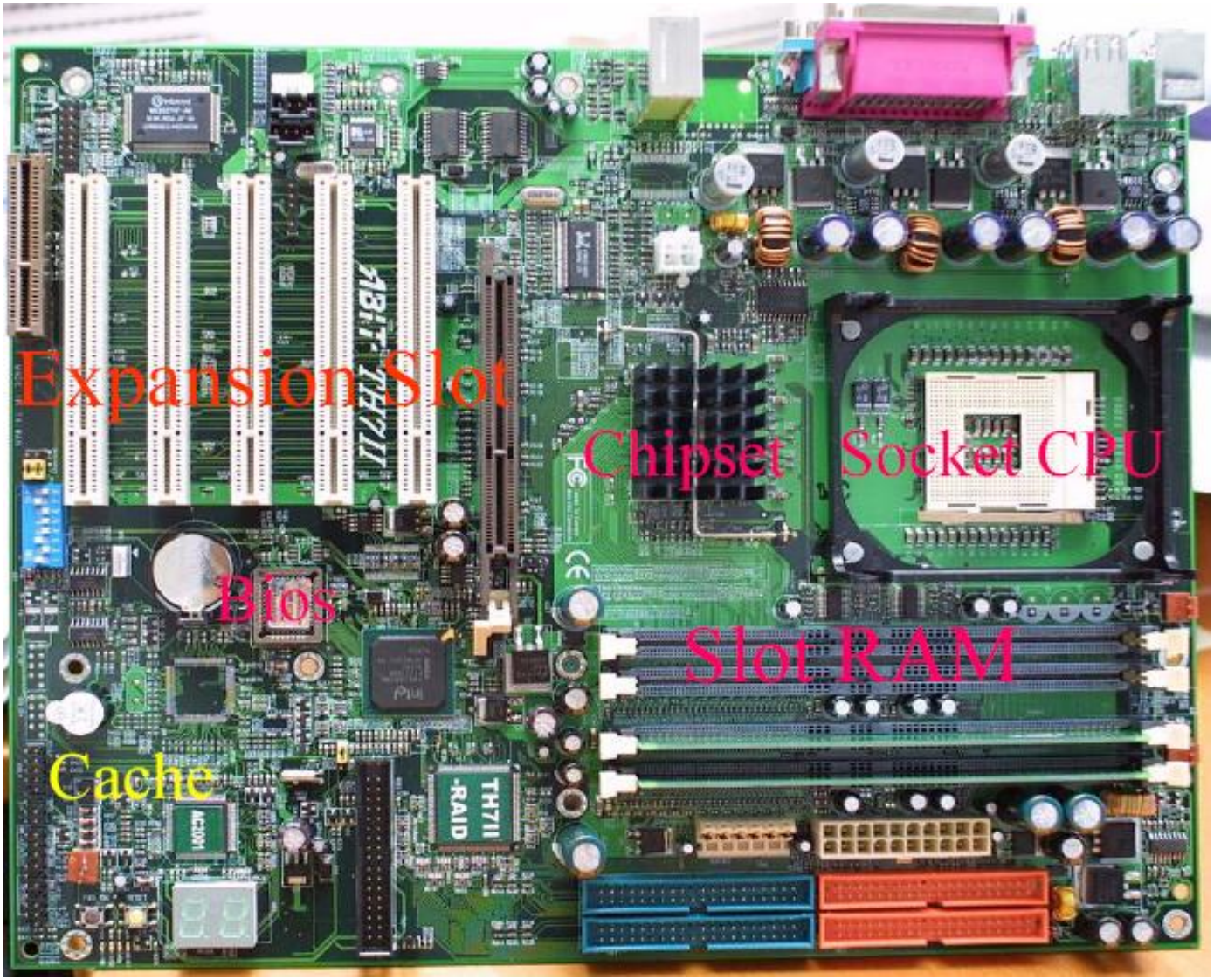
Bản mạch chính là nơi lưu trữ các đường nối giữa các vi mạch, đặc biệt là hệ thống bus.



- Chuẩn AT, ATX, BTX.

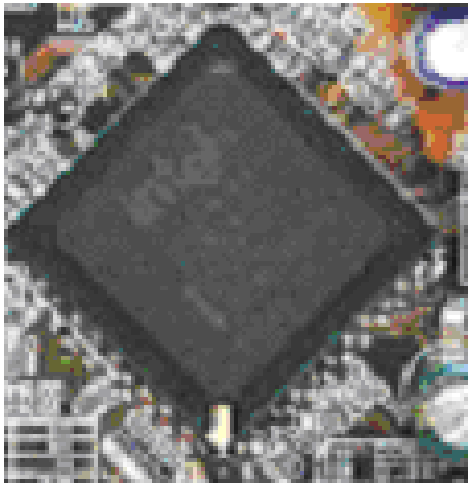
- Các loại Socket: 478, 775, 939…

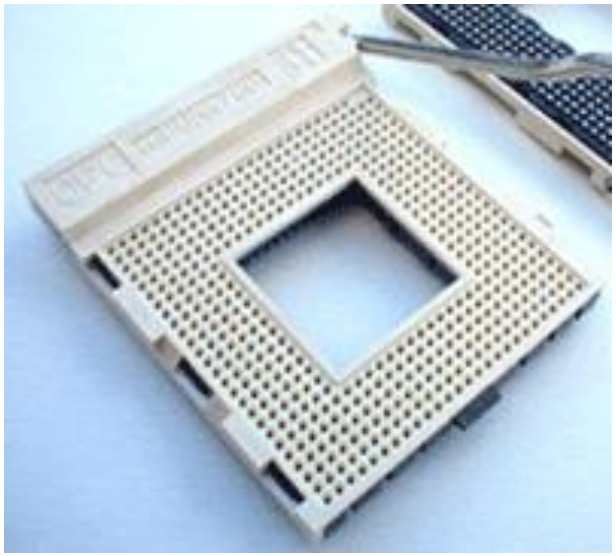
2. Mainboard



2. Mainboard

- ***Form factor***
  - ***ATX:*** 30,5 x 24,4 cm
  - **MiniATX:** 20,8 x 18,5 cm
- ***Chipset***



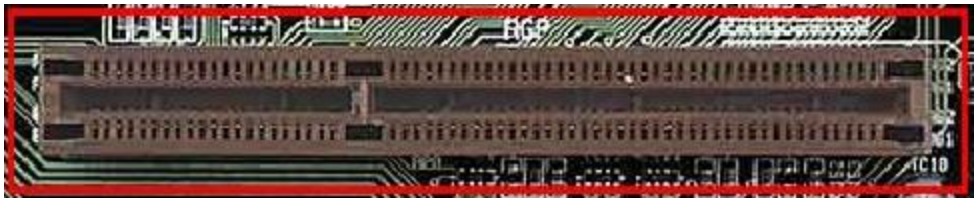


(CPU Socket)

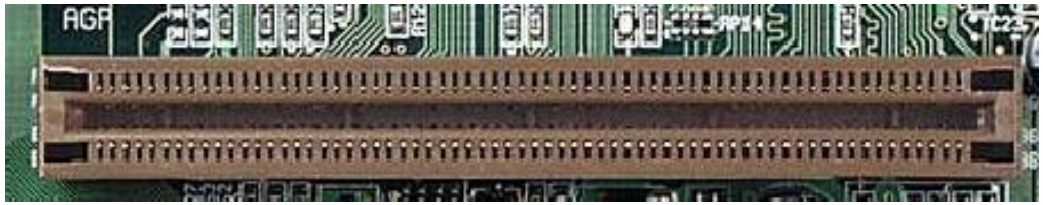
Giao tiếp với CPU:
- Slot
- Socket

2. Mainboard

AGP Slot



AGP	Độ rộng	Xung nhịp	Tốc độ Bytes/Sec	Tốc độ Bits/Sec
1x	32 bits	66MHz	266MB/s	2.1Gbps
2X	32 bits	133MHz	533MB/s	4.3Gbps



Độ rộng	Xung nhịp	Tốc độ Bytes/Sec	Tốc độ Bits/Sec
32 bits	266MHz	1.06GB/s	8.5Gbps

2. Mainboard

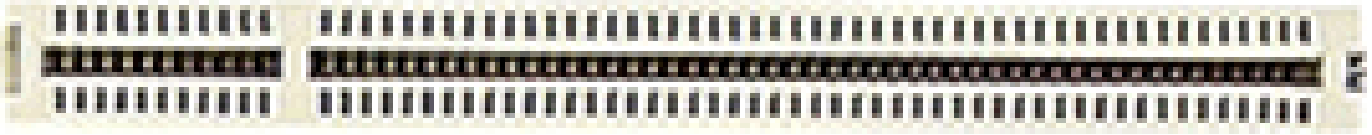
❑ RAM slot

- **Công dụng:** Dùng để cắm RAM và main.
- **Nhận dạng:** Khe cắm RAM luôn có cần gạt ở 2 đầu.
- Lưu ý: Tùy vào loại RAM (SDRAM, DDRAM, RDRAM) mà giao diện khe cắm khác nhau

2. Mainboard

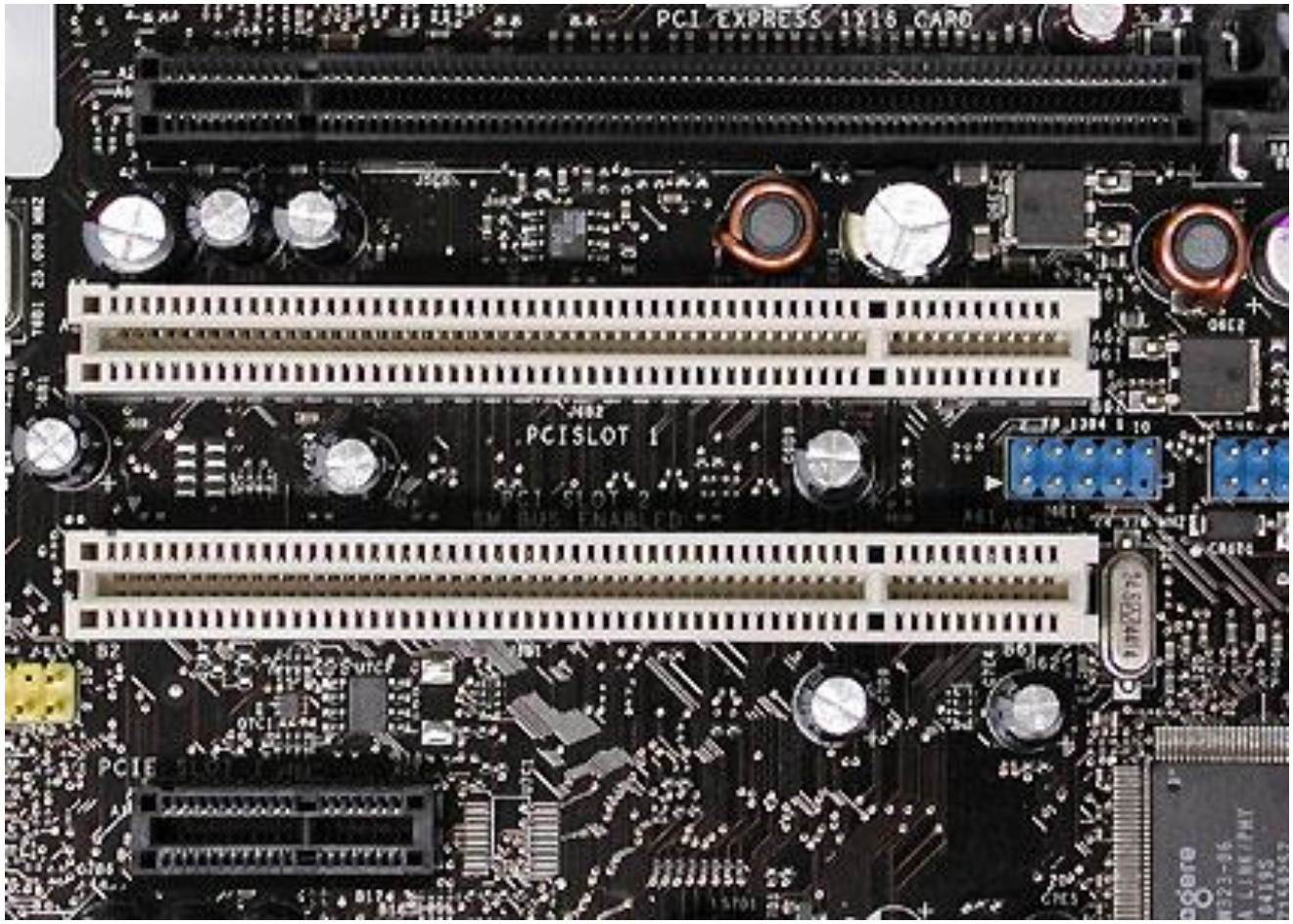
❑ PCI Slot

- **PCI - Peripheral Component Interconnect -** ng
- **Công dụng:** Dùng để cắm các loại card như card mạng, card âm thanh, ...
- **Nhận dạng:** khe màu trắng sử nằm ở phía rìa mainboard.



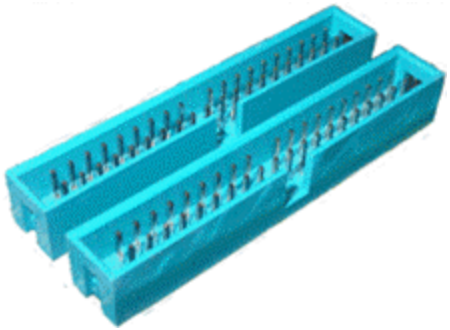
2. Mainboard

- PCI (màu trắng) và PCI Express x16 (màu đen)

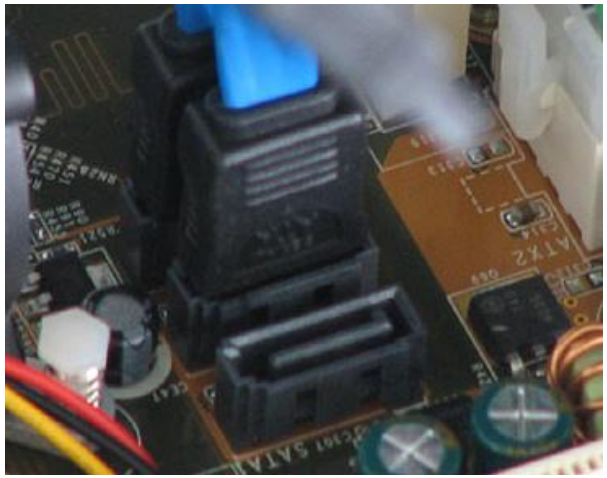


2. Mainboard

- IDE Header
  - **Intergrated Drive Electronics** - 40 chân, cắm các loại ổ cứng, CD
  - **IDE1**: chân cắm chính, để cắm dây cáp nối với ổ cứng chính
  - **IDE2**: chân cắm phụ, để cắm dây cáp nối với ổ cứng thứ 2 hoặc các ổ CD, DVD...



- SATA, SATA2



2. Mainboard

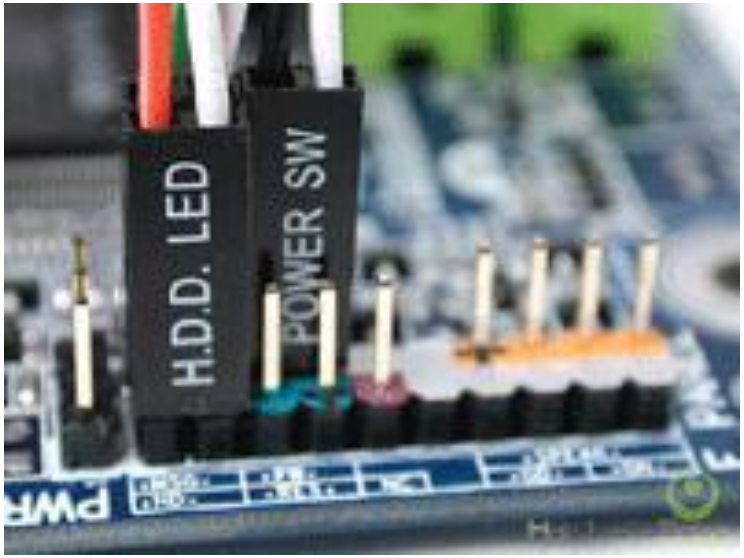
ROM BIOS

- Là bộ nhớ sơ cấp của máy tính. ROM chứa hệ thống lệnh nhập xuất cơ bản (BIOS - Basic Input Output System) để kiểm tra phần cứng, nạp hệ điều hành nên còn gọi là ROM BIOS.

2. Mainboard

- PIN CMOS
  - Là viên pin 3V nuôi những thiết lập riêng của người dùng như ngày giờ hệ thống, mật khẩu bảo vệ ...

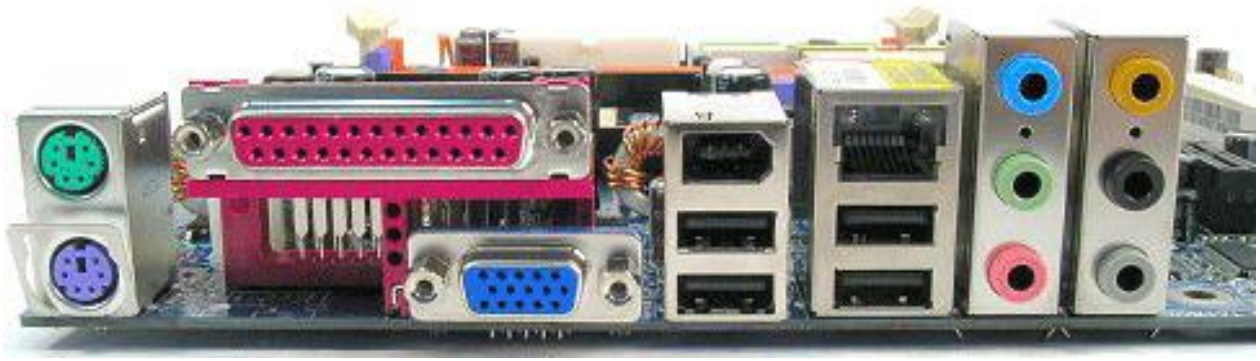
- Power Connector.



Dây nối với vỏ máy (case)

2. Mainboard

- Bên ngoài của mainboard



VÍ DỤ:

Mainboard :ASUS Intel 915GV P5GL-MX, Socket 775/ s/p 3.8Ghz/ Bus 800/ Sound& Vga, Lan onboard/PCI Express 16X/ Dual 4DDR400/ 3 PCI/ 4 SATA/ 8 USB 2.0.

Ví dụ về mainboard

ASUS P5L-VM 1394 *(s/p Core 2 Duo)*	Chip **INTEL 945G** S/P**775 3.8Ghz** Bus 1066, PCI Ex16X,, 2PCI & PCI Ex 1X ATA 100, 4 SataII(3Gb/s), 4DDR2-667, **VGA+ Sound(8ch)+Lan1G Onboard 8USB 2.0, 2IEEE 1394a**	**98**	36 T
ASUS P5B *(s/p Core 2 Duo)*	Chip **INTEL P965**, S/P **775 3.8Ghz**, Bus1333, PCI Ex 16X, 3PCI & 3PCI Ex 1X, ATA 133, 4SataII 3Gb/s, 1Sata II (3Gb) Raid (0,1,JBOD), Ext Sata II (3Gb/s) 4DDR2-800, **Sound(8ch)+ Lan 1G** Onboard, **10USB2.0**	**118**	36 T
ASUS P5K SE *(s/p Core 2 Duo-Core 2 Quad)*	Chip **INTEL P35/ICH9**, S/P **775 3.8Ghz**, Bus1333, 1xPCI Ex 16X, 2PCI & 3xPCI Ex 1X, ATA 133, 4SataII 3Gb/s, Ext Sata II (3Gb/s) 4DDR2-1066(DC), **Sound(8ch)+ Lan 1G** Onboard, **12 USB 2.0**	**110**	36 T

3. Ổ mềm (FDD)

✓Ổ đĩa mềm bao gồm:

- Phần cơ khí.
- Phần điện tử điều khiển động cơ cũng nh□bộ phận đọc/ghi và giải mã.

✓Ổ đĩa phải đảm bảo tốc độ quay chính xác (300 hoặc 360 vòng/phút với sai số 1 đến 2%).

✓Nó còn cần có khả năng định vị đầu từ chính xác(vài micro met) trong thời gian rất ngắn (vài miligiây).

3. Ổ mềm (FDD)

- Có 2 loại đĩa mềm: 5,25 inch và 3,5 inch.
- Cả hai đều có thể tích hợp mật độ ghi thấp (Low Density
- LD), hoặc cao (High Density - HD).

Đặc tính	LD 5,25	HD 5,25	LD 3,5	HD 3,5
Kính thước	5,25	5,25	3,5	3,5
Dung lượng	360Kbyte	1,2 MB	720 Kbyte	1,44MB
Số đường	40	80	80	80
Số sector trong 1 đường	9	15	9	18
Số đầu đọc	2	2	2	2
Số vòng quay/ 1 phút	300	300	300	300
Tốc độ truyền dữ liệu Kbit/s	250	500	250	500

Những thông số chính của 4 loại đĩa mềm

3. Ổ cứng (HDD)

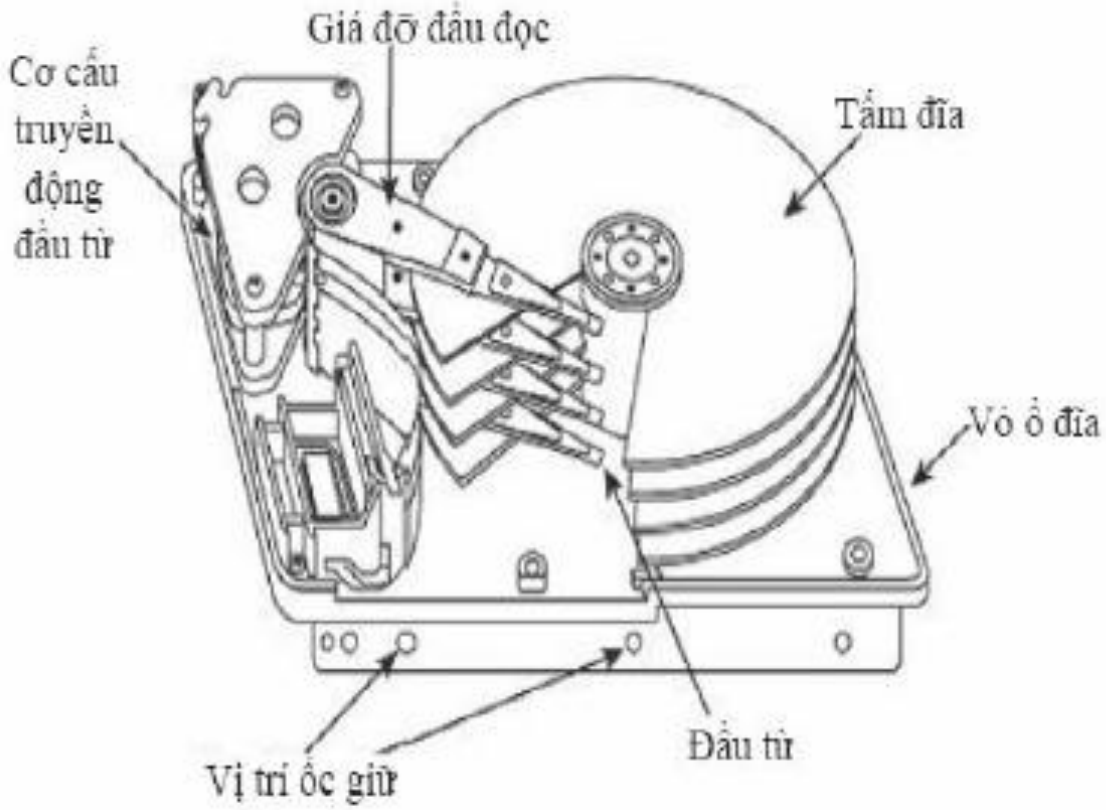
- Đĩa cứng được làm từ vật liệu nền cứng như nhôm, thủy tinh hay gốm.
- Lớp vật liệu nền được phủ một lớp tiếp xúc bám (nickel) phía trên lớp tiếp xúc bám là màng từ lưu trữ dữ liệu (Cobalt).
- Bề mặt trên cùng được phủ một lớp chống ma sát (graphit hay saphia).

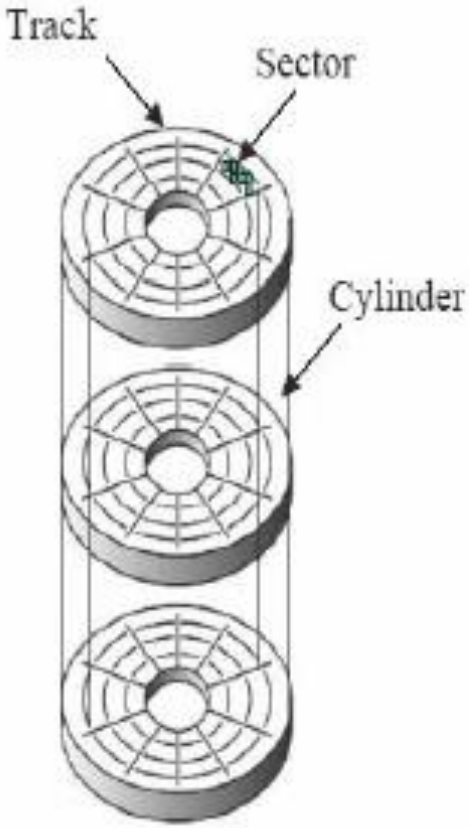
Thời gian truy nhập được phân loại như sau:

- Chậm: t > 40ms,
- Trung bình: 28ms < t < 40ms.
- Nhanh: 18ms < t <28ms.
- Cực nhanh: t < 18ms.

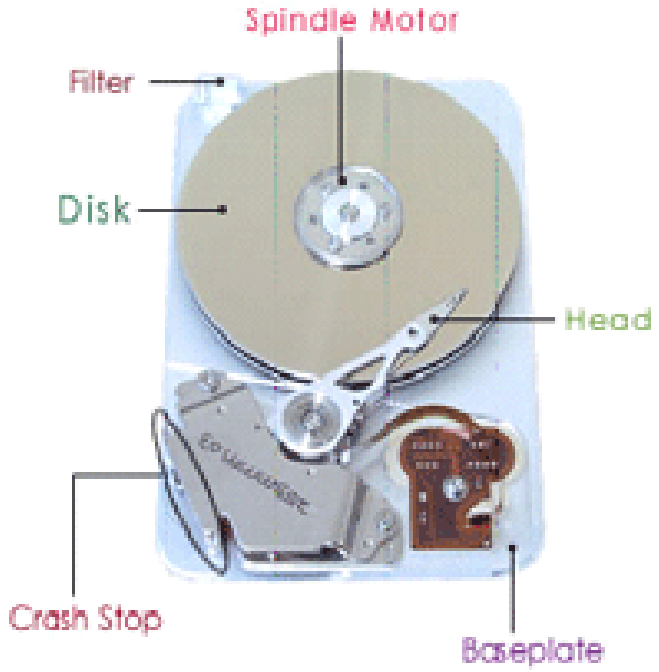
3. Ổ cứng (HDD)

Cấu tạo của HDD





3. Ổ cứng (HDD)



Nguyên tắc hoạt động của đĩa cứng hoàn toàn tương tự đĩa mềm. Điểm khác nhau căn bản là đĩa cứng có dung lượng lưu trữ lớn hơn nhiều so với đĩa mềm.

Các thông số chính:

- Tốc độ quay.
- Dung lượng.
- Tốc độ đọc/ghi.

3. Ổ cứng (HDD)

Các chuẩn giao tiếp đĩa cứng thông dụng

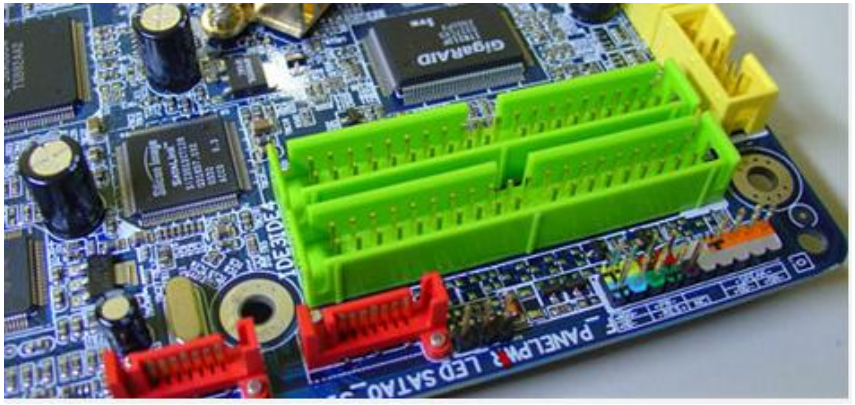
- Intergrated Drive Electronics (**IDE**)
- Small Computer System Interface (**SCSI**)
- **Serial ATA (SATA)**

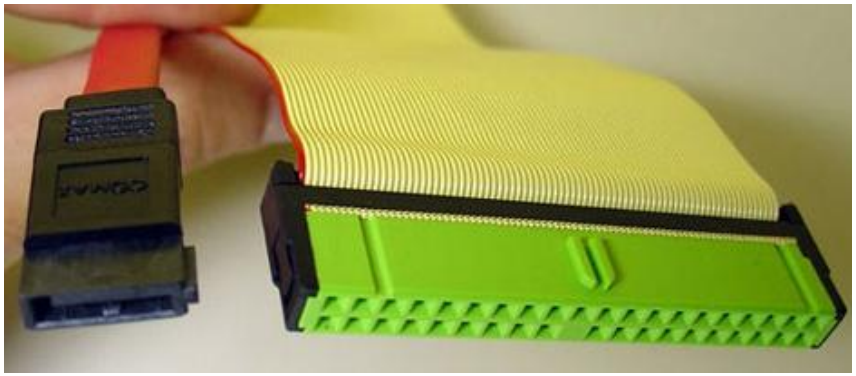
Ví dụ các thông số chính của HDD

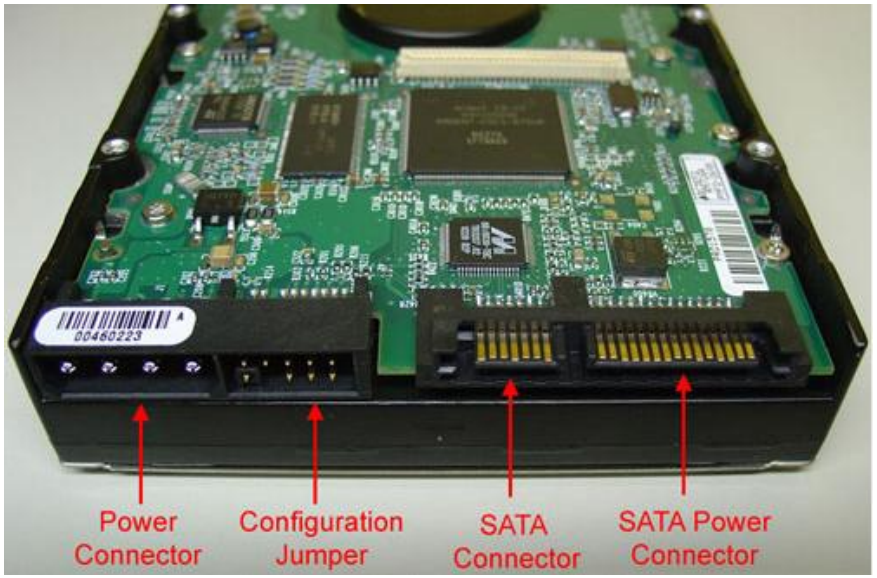
Characteristics	Seagate ST31401N Elite-2 SCSI Drive
Disk diameter (inches)	5.25
Formatted data capacity (GB)	2.8
Cylinders	2627
Tracks per cylinder	21
Sectors per track	≈ 99
Bytes per sector	512
Rotation speed (RPM)	5400
Average seek in ms (random cylinder to cylinder)	11.0
Minimum seek in ms	1.7
Maximum seek in ms	22.5
Data transfer rate in MB/sec	≈ 4.6

3. Ổ cứng (HDD)

Sự khác biệt giữa các loại
HDD IDE (PATA),
SATA (Serial ATA, SATA-150),
SATA II (SATA-300) ?







4. Ổ CD, CDR/W, DVD và DVD R/W

-Thông tin dược lưu trữ trên đĩa quang dưới dạng thay đổi tính chất quang trên bề mặt đĩa.
- Tính chất này được phát hiện qua chất lượng phản xạ một tia sáng của bề mặt đĩa.
- Tia sáng này thường là một tia LASER với bước sóng cố định (790nm đến 850nm).
- Bề mặt đĩa được thay đổi khi ghi để có thể phản xạ tia laser tốt hoặc kém.

- CD-ROM (compact disk read only memory):
- CD-R(RECORDABLE COMPACT DISK)
- CD-WR (writeable/readable compact disk)
- DVD (Digital versatile disc) và DVD R/W

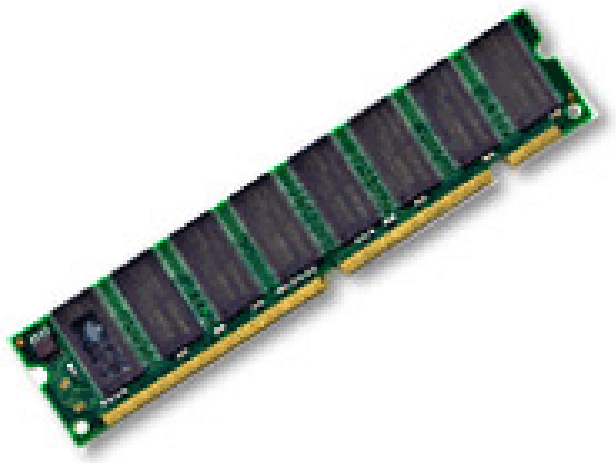
Các tốc độ đọc/ghi: 24X, 32X, 48X, 52X

5. Bộ nhớ RAM và ROM

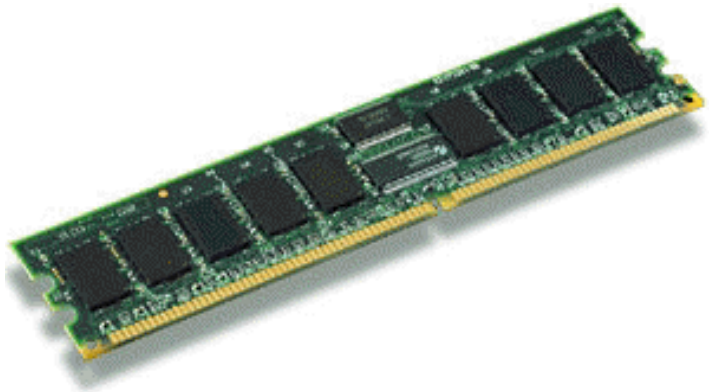
- ***RAM (Random Access Memory)***
  - ***Lưu trữ những chỉ lệnh của CPU, những ứng dụng đang hoạt động, những dữ liệu mà CPU cần***
  - **Đặc trưng:**
    - Dung lượng tính bằng MB, GB.
    - Tốc độ truyền dữ liệu (Bus) tính bằng Mhz.
  - **Phân loại:**
    - Giao diện SIMM - Single Inline Memory Module.
    - Giao diện DIMM - Double Inline Memory Module
- ***Các tế bào nhớ (storage cell)***:
- ***RAM slot***
- ***Interface***

5. Bộ nhớ RAM và ROM

- Các loại DIMM thông dụng hiện nay:
  - **SDRAM (synchronous dynamic random**
    - **Nhận dạng:** SDRAM có 168 chân, 2 khe cắt ở phần chân cắm.
    - **Tốc độ (Bus):** 100Mhz, 133Mhz.
    - **Dung lượng:** 32MB, 64MB, 128MB

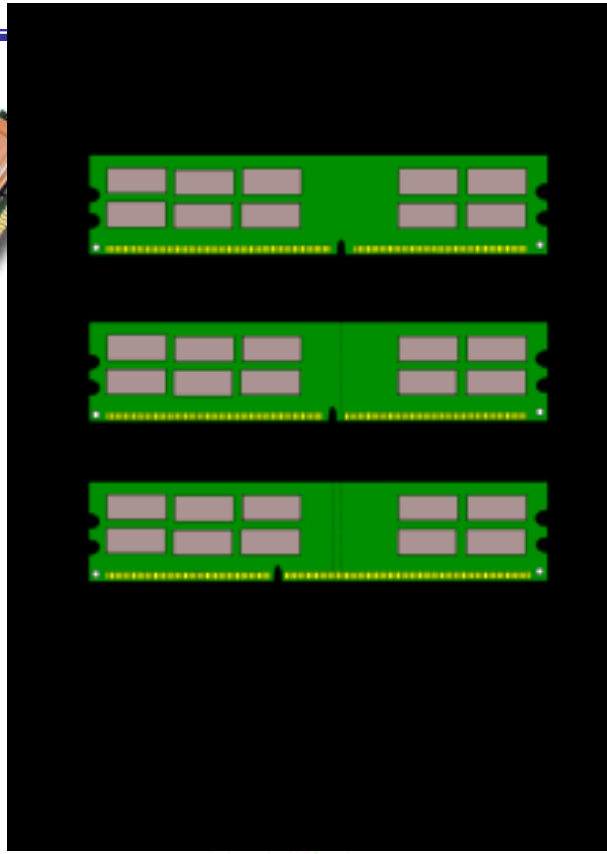


-**DDRAM(**double-data-rate SDRAM)**:**
-184 chân, chỉ có 1 khe cắt ở giữa phần chân cắm.
- Tốc độ (Bus): 266 Mhz, 333Mhz, 400Mhz
- Dung lượng: 128MB, 256MB, 512MB



5. Bộ nhớ RAM và ROM

- DDRAM2
  - Tốc độ gấp đôi DDRAM,
  - không dùng được khe DDRAM.
  - **Tốc độ (Bus):** 400 Mhz
  - **Dung lượng:** 256MB, 512MB



- **RDRAM**
  - 184 chân, có 2 khe cắt gần nhau ở phần chân cắm.
  - bọc tôn giải nhiệt
  - **Tốc độ (Bus):** 800Mhz.
  - **Dung lượng:** 512MB

6. Bàn phím (Keyboard)

Bàn phím (keyboard)

- Thông dụng nhất là các loại MF 101, MF102
- Các cổng bàn phím: COM, PS/2, USB

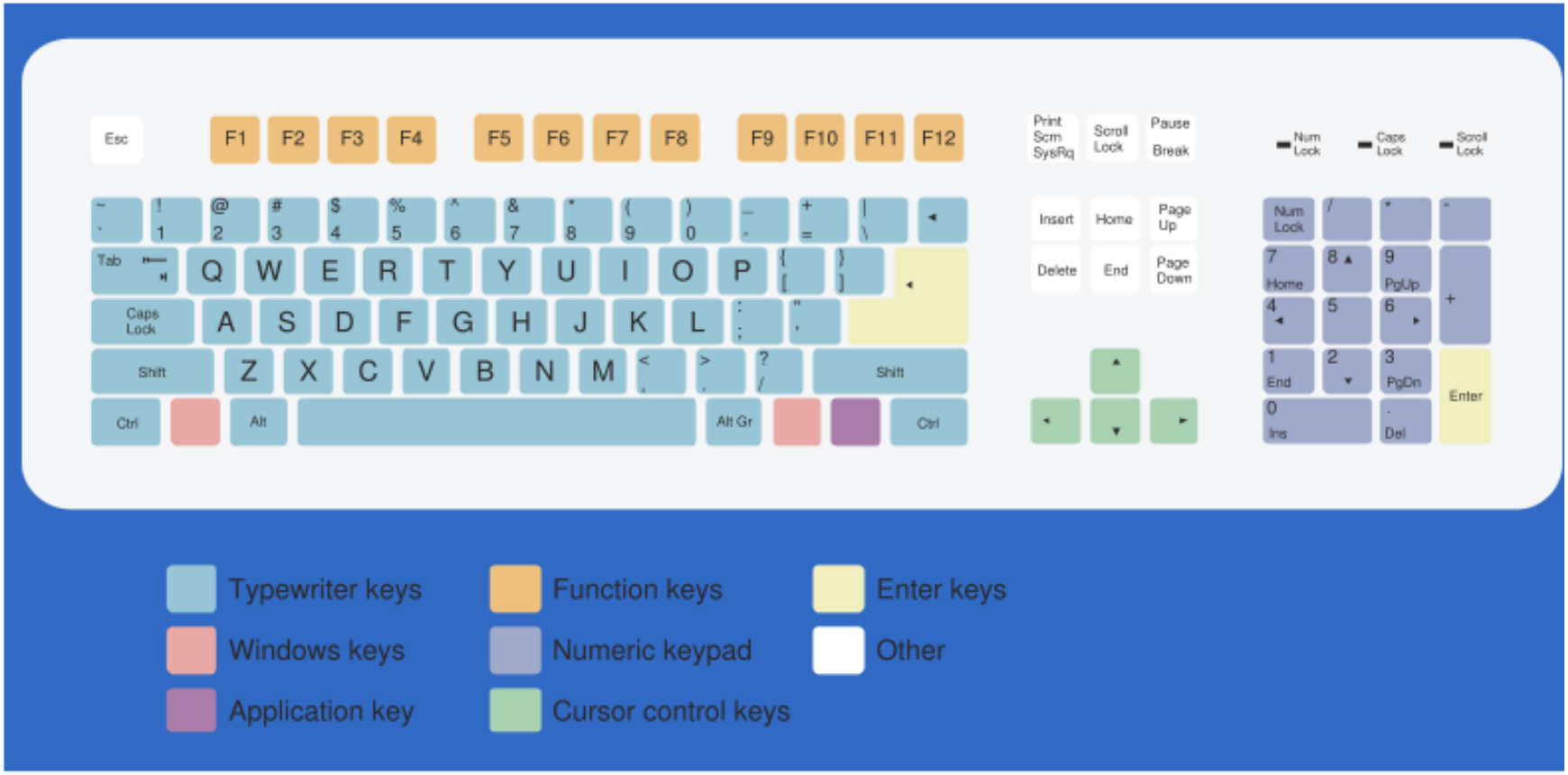
❑ **i:**

– Theo dạng cổng:
  - ng PS/2.
  - ng USB
  - m không dây.

– Theo tính chất:
  - Phím cảm biến điện trở
  - Phím cảm biến điện dung
  - Phím cảm biến điện từ

6. Bàn phím (Keyboard)

- Bàn phím chuẩn của Microsoft



7. Chuột (mouse)

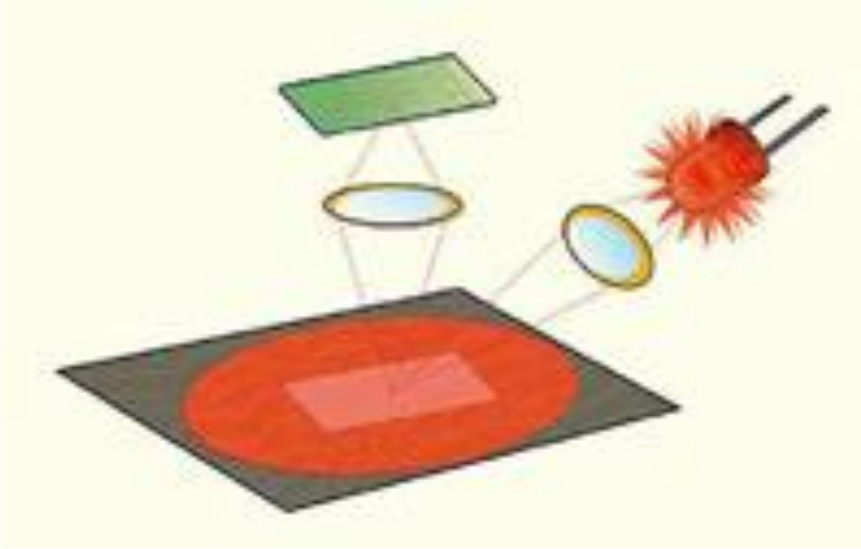
- Phân loại
  - Theo nguyên lý:
    - cơ,
    - quang
    - cơ quang
  - Theo cổng giao tiếp:
    - LPT,
    - COM,
    - PS/2,
    - USB
    - Không dây
      - Sóng radio
      - Bluetooth
      - RFID

Chuột cơ

7. Chuột (mouse)

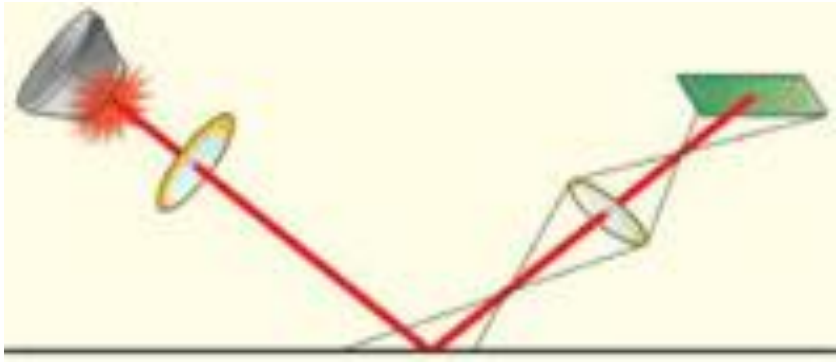
❑ Chuột quang

– Nguyên lý hoạt động



Chuột laser:

- Giới thiệu năm 2004
- Logitech MX1000



8. Card màn hình (VGA Card)

Các mốc lịch sử

Năm	Chuẩn	Ý nghĩa	Kích thước	Số màu
1981	CGA	Colour Graphics Adaptor	640 x 200 160 x 200	Không, 16
1984	EGA VGA	Enhanced Graphics Adaptor Video Graphics Array	640 x 350 640 x 480 320 x 200	6 16 256
1987	XGA	Extended Graphics Array	800 x 600 1024x768	16.7 triệu 65536
1990	SXGA UXGA	Super Extended Graphics Array Ultra XGA	1280x 1024 1600 x 1200	65,536 65,536

8. Card màn hình (VGA Card)

- Dung lượng bộ nhớ video và khả năng hiển thị màn hình

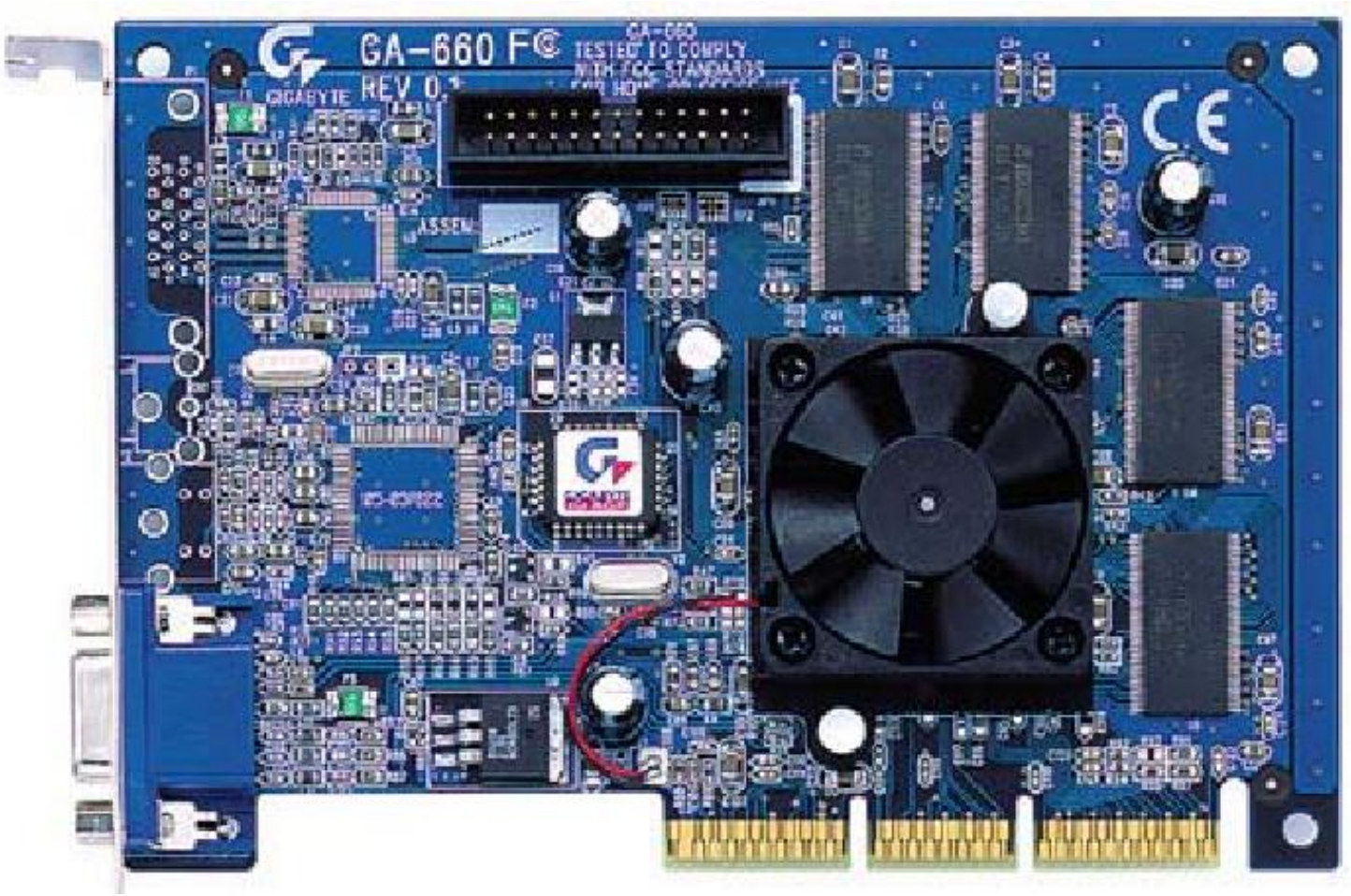
Dung lượng bộ nhớ	Kích thước màn hình	Chiều sâu màu	Số màu
1 Mb	1024x768	8-bit	256
	800 x 600	16-bit	65,536
2Mb	1024 x 768	8-bit	256
	1284 x 1024	16-bit	65,536
	800x600	24-bit	16.7 million
4Mb	1024x768	24-bit	16.7 million
6Mb	1280x1024	24-bit	16.7 million
8Mb	1600x1200	32-bit	16.7 million

8. Card màn hình (VGA Card)

- Bộ gia tốc (***Accelerator***)
- Bộ vi xử lý đồ họa **GPU(Graphics Processing Unit)**
- Bộ nhớ video (VRAM)
- Bộ Chuyển số/tương tự RAMDAC

8. Card màn hình (VGA Card)

- VGA AGP 4x



8. Card màn hình (VGA Card)

- Công ty sản xuất GPU
  - ATI (AMD), nVIDIA
- Các thông số
  - VRAM: 64MB?128MB?
  - DVI-out, S-Video out/in, Firewire (IEEE 1394)
  - Dual display hay Dual head
  - **Quadro Plex 1000 của Nvidia được bán với giá 18.000USD**

128MB ASUS EAX1300HM TD/ 512	Ati RX1300	16X	64bit	DVI, Out TV	PCI Express	**57**
256MB ASUS EN7300GS/HTD	Geforce 7300GS	16X	64bit DDR2	DVI, Out TV	PCI Express	**76**
256MB ASUS EAX1550/TD	Radeon RX1550	16X	128bit DDR2	DVI, Out TV	PCI Express	**75**

9. Màn hình (Monitor)

- Các loại màn hình:
  - Màn hình tia âm cực (CRT - cathode ray tube),
  - Màn hình tinh thể lỏng (LCD - liquid crystal display),
  - Màn hình plasma (PD - plasma display),
  - Màn hình công nghệ mới:LED, Laser, SED, OLED
- Các đặc tính chung:
  - **Vùng hiển thị hình ảnh (Viewable area)**
  - **Độ phân giải** (**Resolution**)- *Số lượng các điểm trên trục ngang và dọc*

    640x480, 1024x768, 1280x1024
  - **Mật độ điểm ảnh** - *số điểm ảnh trên một đơn vị chiều dài (dpi - dot per inch*).

9. Màn hình (Monitor)

❑ Các đặc tính chung (tt):

- **Khoảng cách giữa tâm các điểm ảnh (Dot pitch):** 0.28mm, 0.27mm, 0.26mm, 0.25mm,…
- **Độ sâu của màu (Colour Depth):** 16,8 triệu màu, 65.000 màu,…
- **Tốc độ làm tươi hình ảnh hay tần số quét của màn hình (Refresh Rate):** 50 Hz, 60 Hz, 72 Hz, 85 Hz, 90 Hz, 100 Hz…).
- **Tỉ số giữa chiều rộng và chiều cao (Respect ratio):** 4:3
- **Power Consumption:** công suất tiêu thụ điện của màn hình

❑ Độ phân giải được phân loại như sau:

- Phân giải thấp (<50 dpi).
- Phân giải trung bình (51dpi - 70dpi).
- Phân giải cao (71dpi - 120dpi).
- Phân giải siêu cao (>l20 dpi)

9. Màn hình (Monitor)

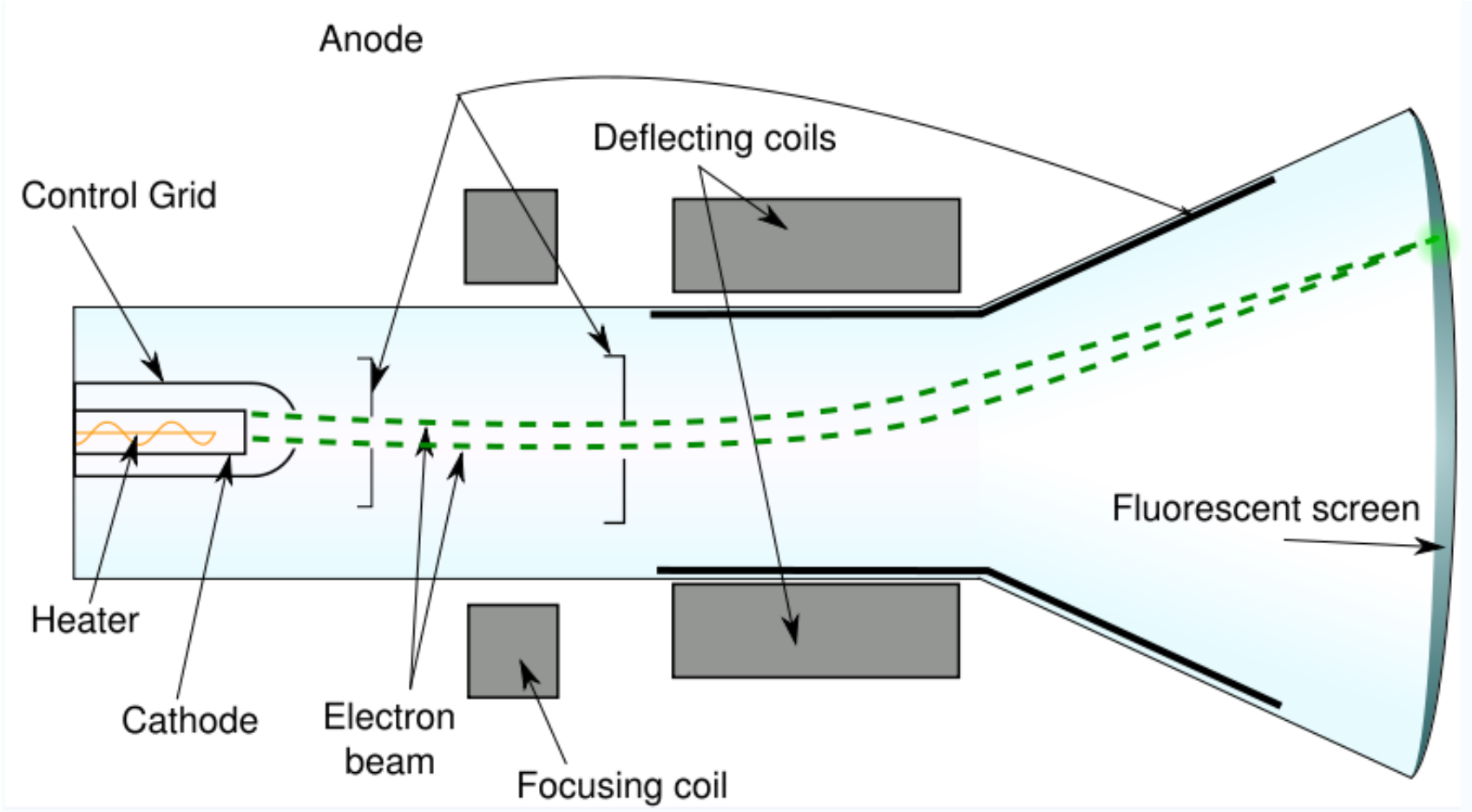
Kích thước màn hình thường là 640x480, 800x600, 1024x768, 1280x1024,… .

Một màu bất kỳ có thể biểu diễn qua **ba màu cơ bản: đỏ, xanh lục, xanh nước biển** tuỳ theo độ đậm nhạt (gray scale). Độ sâu màu (color depth) là số màu có thể hiển thị được cho một điểm ảnh. Tuỳ theo số bit được dùng để hiển thị màu ta phân loại màn hình theo màu như sau:

- Đen trắng 1 bit (2 màu),
- Màu CGA 4 bit (16 màu),
- Màu giả (pseudo color) 8 bit (256 màu),
- Màu cao(high color) 16 bit,
- Màu thật (true color) 24 bit
- Màu siêu thật (highest color) 32 bit

9. Màn hình (Monitor)

❑ Cấu tạo màn hình CRT



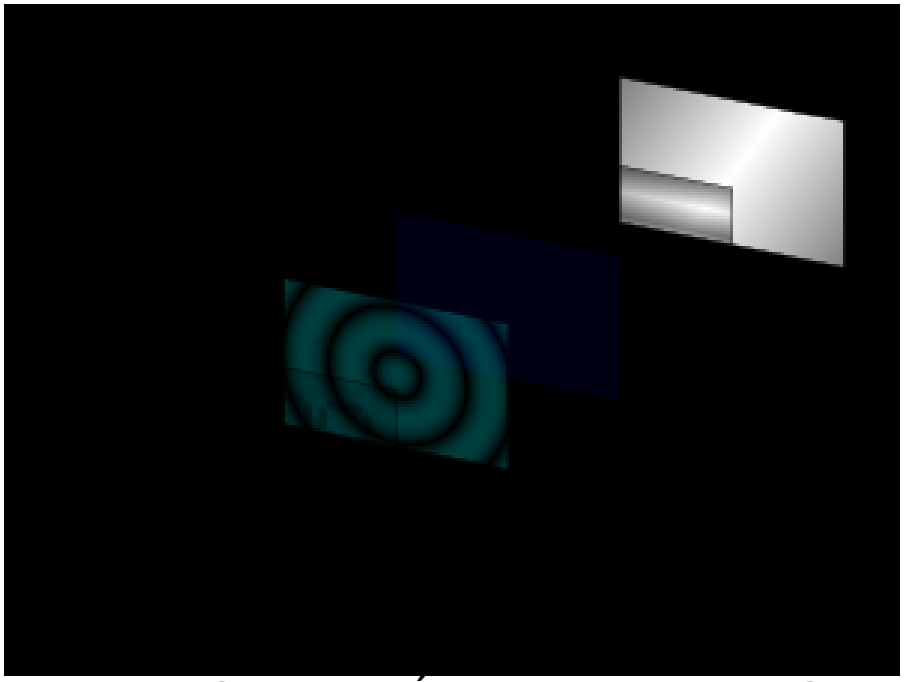
9. Màn hình (Monitor)

- **Ánh sáng phân cực:** **Ánh sáng phân cực là ánh sáng chỉ có một phương dao động duy nhất, gọi là phương phân cực.**
- **Kính lọc phân cực:** là loại vật liệu chỉ cho ánh sáng phân cực đi qua. Lớp vật liệu phân cực có một phương đặc biệt gọi là quang trục phân cực
- **Tinh thể lỏng:**
  - Không có cấu trúc mạng tinh thể cố định như các vật rắn,
  - Các phân tử có thể chuyển động tự do trong một phạm vi hẹp như một chất lỏng.
  - Các phân tử trong tinh thể lỏng liên kết với nhau theo từng nhóm
  - Giữa các nhóm có sự liên kết và định hướng nhất định,
  - Cấu trúc của chúng có phần giống cấu trúc tinh thể.

9. Màn hình (Monitor)

❑ Cấu tạo màn hình LCD

- 1-lớp kính lọc phân cực có quang trục phân cực dọc,
- 2-tấm thủy tinh mỏng,
- 3-lớp tinh thể lỏng,
- 4-tấm thủy tinh mỏng,
- 5-lớp kính lọc phân cực có quang trục phân cực ngang,
- 6- lớp đèn nền, cung cấp ánh sáng nền



Các lớp cấu tạo màn hình LCD

Cường độ sáng của điểm ảnh phụ thuộc vào lượng ánh sáng truyền qua kính lọc phân cực thứ hai. Lượng ánh sáng này lại phụ thuộc vào góc giữa phương phân cực và quang trục phân cực. Góc này lại phụ thuộc vào độ xoắn của các phân tử tinh thể lỏng. Độ xoắn của các phân tử tinh thể lỏng phụ thuộc vào điện áp đặt vào hai đầu tinh thể lỏng

9. Màn hình (Monitor)

Các điểm cần chú ý khi mua LCD

- 1. Kích thước màn hình (screen size): 17” CRT ~ 15” LCD.
- 2. Độ phân giải (resolution)
- Tốc độ làm tươi (refresh rate): 60Hz, 75Hz
- Tần số đáp ứng (response rate): 25ms = 9ms (rising) + 16ms (falling), 20ms, 16ms.
- Độ tương phản (contrast):200:1 tới 700:1
- Góc nhìn (viewing angles): theo chiều dọc và ngang, nên chọn trên 160 độ
- Giao tiếp tương tự (D-Sub) và giao tiếp số (DVI)
- Độ sáng (brightness): 50-60 đến 100%

10. Card mạng & Modem

Card mạng

(Network adapter)

Dùng để kết nối 1 máy tính vào 1 mạng LAN

Modem

Kết nối máy tính với Internet thông qua đường dây điện thoại

Ví dụ máy tính bộ & notebook

ROBO VICTOR-VB 12.130.000 VNĐ
- Mainboard chipset Intel P35, Sound & Lan Onboard, Bus 1333, USB 2.0, PCI Express, 4 SATA-II, 4 DDR2-1066.
- CPU Intel Core Quad E4400 (2.0GHz).
- DDR-II 1GB x 2 = 2GB - Bus 667.
- HDD 250.0GB SATA-II (ATA/300, 7200rpm).
- VGA Palit Geforce 8600GT 256MB/DDR3
- Monitor LCD 15". - Case ROBO ATX 450W.
- Keyboard & Mouse ROBO PS/2.
- DVD COMBO 16X.

Acer 5573 AWXMi	LX.AXMOC.026 Centrino Core Duo T2350 2x1.86Ghz	DDRII 1GB 160GB SATA	128MB Gforce 5.1 Reader	DVD RW Webcam
14” WXGA Mirror	10/100 56K -WL	2.4Kg 6cell Bluetooth Finger print Webcam	Vista Home Pre	**999**

Chương 3
Biểu diễn dữ liệu

3.1. Khái niệm thông tin
3.2. Lượng thông tin và sự mã hóa thông tin
3.3. Hệ thống số
3.4. Các phép tính số học cho hệ nhị phân
3.5. Số quá n (excess-n)
3.6. Cách biểu diễn số với dấu chấm động
3.7. Biểu diễn số BCD
3.8. Biểu diễn các ký tự

Mục tiêu

- Hiểu các hệ cơ số thông dụng và cách chuyển đổi.
- Hiểu phương pháp biểu diễn số nguyên và số chấm động.
- Hiểu các phương pháp tính đơn giản với các số.
- Hiểu các phương pháp biểu diễn số BCD và ký tự

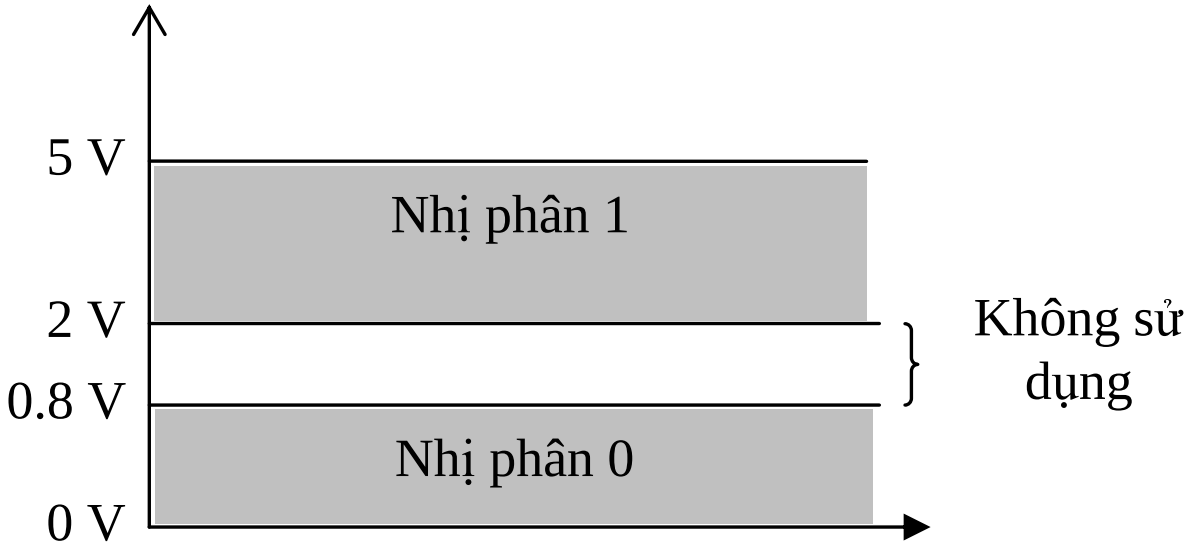
Hình dung về "biểu diễn dữ liệu"

- Mọi thứ trong máy tính đều là 0 và 1
- Thế giới bên ngoài có nhiều khái niệm như con số, chữ cái, hình ảnh, âm thanh,…

→ biểu diễn dữ liệu = quy tắc "gắn kết" các khái niệm trong thế giới thật với một dãy số 0 và 1 trong máy tính

3.1. Khái niệm thông tin

- Dùng các tín hiệu điện thế
- Phân thành các vùng khác nhau



Hình 3.1. Biểu diễn trị nhị phân qua đ

3.2. Lượng thông tin và sự mã hoá thông tin

- Thông tin được đo lường bằng đơn vị thông tin mà ta gọi là bit.
- Lượng thông tin được định nghĩa bởi công thức:

$$I = Log_2(N)$$

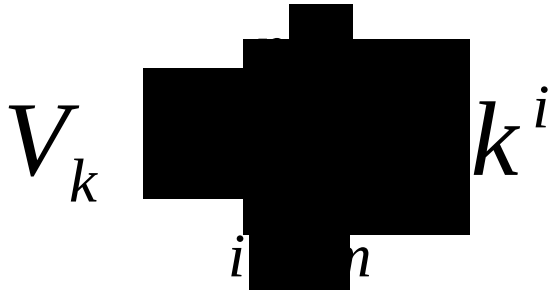
  - Trong đó:
    - I: là lượng thông tin tính bằng bit
    - N: là số trạng thái có thể có
  - Ví dụ, để biểu diễn một trạng thái trong trạng thái có thể có, ta cần một số bit ứng với một lượng thông tin là:

$$I = Log_2(8) = 3 \text{ bit}$$

Trạng thái	*A2*	*A1*	*A0*
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

3.3. Hệ Thống Số

- Dạng tổng quát để biểu diễn giá trị của một số:



  - Trong đó:
    - V_k: Số cần biểu diễn giá trị
    - *m*: số thứ tự của chữ số phần lẻ (phần lẻ của số có m chữ số được đánh số thứ tự từ -1 đến -m)
    - *n-1*: số thứ tự của chữ số phần nguyên (phần nguyên của số có n chữ số được đánh số thứ tự từ 0 đến n-1)
    - b_i: giá trị của chữ số thứ i
    - *k*: hệ số (k=10: hệ thập phân; k=2: hệ nhị phân;...).

3.3. Hệ Thống Số

❑ Các hệ đếm (cơ số) thông dụng

- **Thập phân (Decimal)**
  - 10 chữ số : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

2	1	0		-1	←trọng số
2	**3**	**5**	**.**	**3**	$= 2*10^2 + 3*10^1 + 5*10^0 + 3*10^{-1}$

- **Nhị phân (Binary)**
  - 2 chữ số: 0, 1
  - Ví dụ số m = 1101,011 ở hệ nhị phân biểu diễn một đại lượng:

$$m_2 = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3}$$

- **Bát phân (Octal)**
  - 8 chữ số: 0, 1, 2, 3, 4, 5, 6, 7
- **Thập lục phân (Hexadecimal)**
  - 16 chữ số: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E. F
    - A=10, B=11, C=12, D=13, E=14, F=15

Các hệ đếm (cơ số) thông dụng

HỆ ĐẾM	CƠ SỐ	KÍ HIỆU CHỮ SỐ	TRỌNG SỐ	VÍ DỤ
Nhị phân	2	0, 1	2^i	1001,1101
Bát phân	8	0,1,2,3,4,5,6,7	8^i	3567,24
Thập phân	10	0,1,2,3,4,5,6,7,8,9	10^i	1369,354
Thập lục phân	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	16^i	3FA9,6B

Bảng 3.2. Các hệ đếm cơ bản

Chuyển đổi từ cơ số 10 sang b

Quy tắc:

Chia số cần đổi cho b, lấy kết quả chia tiếp cho b cho đến khi **kết quả** bằng **0**. Số ở cơ số b chính là các số dư (của phép chia) viết ngược.

Ví dụ:

41	2	= 20	dư	1
20	2	= 10	dư	0
10	2	= 5	dư	0
5	2	= 2	dư	1
2	2	= 1	dư	0
1	2	= 0	dư	1

$41_{10} = 101001_2$

Chuyển đổi hệ 10 sang Nhị phân

Quy tắc: Người ta chuyển đổi từng phần nguyên và lẻ theo quy tắc sau

Phần nguyên: Chia liên tiếp phần nguyên cho 2 giữ lại các số dư, Số nhị phân được chuyển đổi sẽ là dãy số dư liên tiếp tính từ lần chia cuối về lần chia đầu tiên.

Phần lẻ: Nhân liên tiếp phần lẻ cho 2, giữ lại các phần nguyên được tạo thành. Phần lẻ của số Nhị phân sẽ là dãy liên tiếp phần nguyên sinh ra sau mỗi phép nhân ***tính từ lần nhân đầu đến lần nhân cuối***

Chuyển đổi hệ 10 sang Nhị phân

Ví dụ: **Chuyển sang hệ Nhị phân số: 13,6875**

Thực hiện:

Phần nguyên: 13:2 = 6 dư 1
6:2 = 3 dư 0
3:2 = 1 dư 1
1:2 = 0 dư 1

Phần nguyên của số Nhị phân là 1101

Phầnlẻ:

0,6875 x 2 = 1,375 Phần nguyên là 1
0,375 x 2 = 0,750 Phần nguyên là 0
0,750 x 2 = 1,500 Phần nguyên là 1
0,5 x 2 = 1,00 Phần nguyên là 1

Phần lẻ của số Nhị phân là: 0,1011

Ta viết kết quả là: $(13,6875)_{10} = (1101,1011)_2$

Chuyển đổi từ cơ số 10 sang b

- **Quy tắc:** Chia số cần đổi cho b, lấy kết quả chia tiếp cho b cho đến khi **kết quả** bằng **0**. Số ở cơ số b chính là các số dư (của phép chia) viết ngược.
- Ví dụ:

41 16 = 2 dư 9
2 16 = 0 dư 2

$41_{10} = 29_{16}$

Chuyển đổi từ cơ số 10 sang b

Ví dụ: Chuyển số $(3287,5100098)_{10}$ sang Cơ số 8.

❑ Phần nguyên:

3287:8 = 410	dư	↑	7
410:8 = 51	dư		2
51:8 = 6	dư		3
6:8 = 0	dư		6

Vậy $(3287)_{10}=(6327)_8$

❑ Phần lẻ:

0,5100098x8 = 4,0800784	phần nguyên là 4	↓
0,0800784x8= 0,6406272	phần nguyên là 0	
0,6406270x8= 5,1250176	phần nguyên là 5	
0,1250176x8= 1,0001408	phần nguyên là 1	

Vậy $(0,5100098)_{10}=(0,4051)_8$

Kết quả chung là: $(3287,5100098)_{10}=(6327,4051)_8$

Chuyển đổi từ cơ số b sang 10

- Việc chuyển đổi từ một hệ cơ số bất kỳ sang hệ 10 thì đơn giản hơn và cách làm như trong trường hợp định nghĩa đại lượng của số đó.
- ***Ví dụ***: $235,3_8$ -> hệ 10

2	1	0		-1	←trọng số
2	**3**	**5**	**.**	**3**	$= 2*8^2 + 3*8^1 + 5*8^0 + 3*8^{-1} = 157.375_{10}$

Chuyển đổi hệ 2 sang hệ 10

Ví dụ: Chuyển đổi sang hệ Thập phân số: **m = 1101,011**

Thực hiện: Ta lập tổng theo trọng số của từng Bit nhị phân:

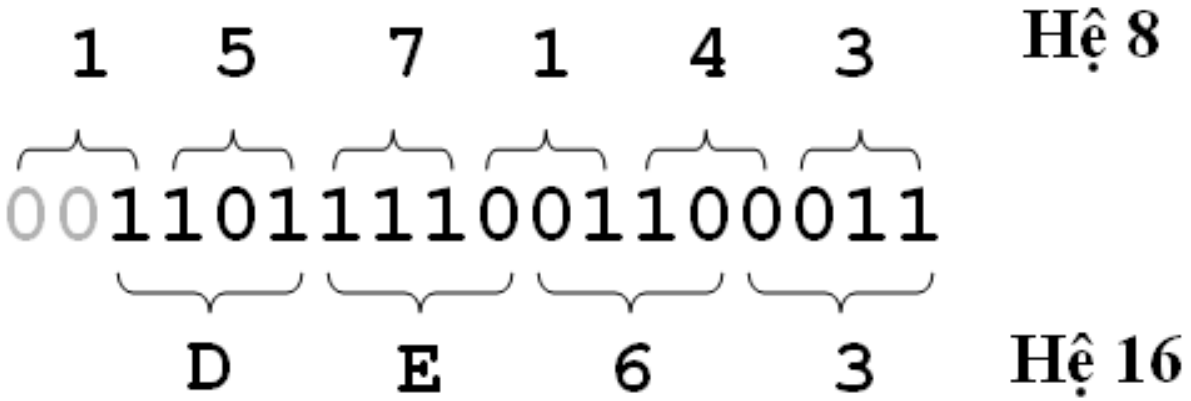
$$m = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3}$$

$$m = 8 + 4 + 0 + 1 + 0 + 1/4 + 1/8$$

$$m = 13,375$$

Chuyển đổi cơ số 2-8-16

- **Quy tắc:** Từ phải sang trái, gom **3** chữ số nhị phân thành một chữ số **bát phân** hoặc gom **4** chữ số nhị phân thành một chữ số **thập lục phân**



Chuyển đổi cơ số 2-8-16

Ví dụ: Chuyển số M = $(574,321)_8$ sang biểu diễn nhị phân.

Thực hiện: Thay mỗi chữ số bằng nhóm nhị phân 3 bit tương ứng:

M =	101	111	100	,	011	010	001
	5	7	4		3	2	1

Ví dụ: Chuyển số M = $(1001110,101001)_2$ sang cơ số 8.

Thực hiện:

M =	1	001	110	,	101	001
M =	1	1	6	,	5	1

M = $(116,51)_8$

Tương quan giữa các hệ thống số

Hệ 2 (Base 2)	Hệ bát phân (Base 8)	Hệ thập phân (Base 10)	Hệ thập lục phân (Base 16)
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Các phép tính số học cho hệ nhị phân

❑ *Phép cộng hai số nhị phân không dấu*

- Khi cộng, thực hiện từ bit có trọng số thấp đến bit có trọng số cao.
- Nếu có số nhớ thì số nhớ sinh ra được cộng vào bit có trọng số cao hơn liền kề

SỐ HẠNG 1	SỐ HẠNG 2	TỔNG	SỐ NHỚ	KẾT QUẢ
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	10

Bảng 3.4. Quy tắc Cộng Nhị phân cho 2 số 1 bit.

Phép trừ hai số nhị phân không dấu

- Phép tính được thực hiện từ Bit có trọng số thấp đến Bit có trọng số cao.
- Số vay sẽ được trừ vào Bit có trọng số cao hơn ở liền kề.

SỐ BỊ TRỪ	SỐ TRỪ	HIỆU SỐ	SỐ VAY
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Bảng 3.5. Quy tắc trừ Nhị phân cho 2 số 1 bit.

Phép nhân và chia hai số nhị phân không dấu

- ***Phép nhân*** nhị phân được thực hiện như nhân thập phân

Ví dụ: Có phép tính: 1001 nhân với 1101

Ta thực hiện:	1001	(Số bị nhân-Multiplicant)
x	1101	(Số nhân-Multiplier)
	1001	
	0000	
+	1001	
	1001	
Kết quả là:	1110101	

Phép nhân và chia hai số nhị phân không dấu

- ***Phép chia*** nhị phân được thực hiện như chia thập phân

Ví dụ: Có phép tính: 1110101 chia cho 1001

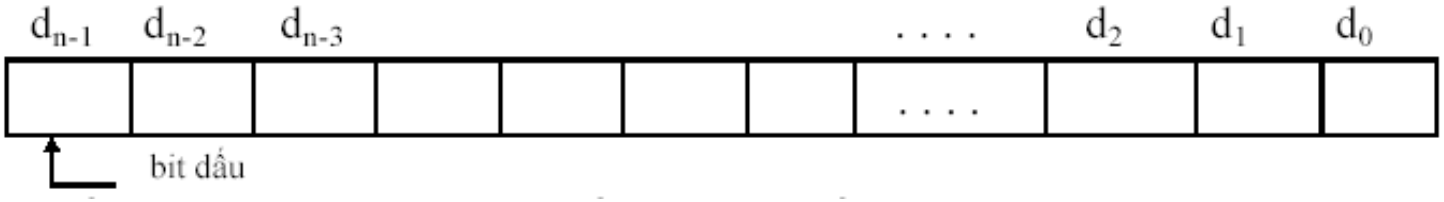
Ta thực hiện:	1110101	:	1001
-	1001		1101
	01011		
-	1001		
	001001		
	1001		
	0000		
Kết quả:	1111010	:	1001 = 1101

Biểu diễn số nguyên có dấu

- Có ba cách để biểu diễn một số nguyên n bit có dấu
  - Biểu diễn bằng trị tuyệt đối và dấu
  - Biểu diễn bằng số bù 1
  - Biểu diễn bằng số bù 2

Biểu diễn số nguyên có dấu

- Biểu diễn bằng trị tuyệt đối và dấu



- $+25_{10} = \mathbf{0}0011001_2$
- $-25_{10} = \mathbf{1}0011001_2$
- Một Byte (8 bit) có thể biểu diễn các số có dấu từ -127 tới +127.
- Có hai cách biểu diễn số không là 0000 0000 (+0) và 1000 0000 (-0).

Số bù

- Quy tắc chung (r: cơ số, n: số chữ số)
  - Bù (r-1) của N = (r^n – 1) – N
  - Bù r của N = r^n – N
    - Bù r của (bù r của N) = N
    - **Nhận xét:** Có tính chất giống – (– N) = N
- **Đối với hệ 10**
  - Bù 9 của *N* = 9 - từng ký số
    - VD: Bù 9 của 43520 là 99999 – 43520 = 56479
  - Bù 10 của N = bù 9 +1
    - VD: bù 10 của 43520 là 56479+ 1 = 56480
    - Mẹo: Bù 10 của 347200 là 652800

Số bù (tt)

- Đối với hệ nhị phân:
  - Bù 1 = đảo n bit của N
    - Bù 1 của (1100) = 0011
  - Bù 2 = bù 1 + 1
    - Bù 2 của (1100) = 0011 + 1 = 0100
    - Mẹo: giữ nguyên các số 0 bên phải cho đến khi gặp số 1, sau đó đảo

      1100

      0100

Biểu diễn số nguyên có dấu ở dạng bù 1

- Đối với số dương thì biểu diễn giống dấu và trị tuyệt đối
- Đối với số âm thì được biểu diễn dưới dạng bit dấu và giá trị của số đó ở dạng bù 1. Ta cũng có thể hiểu là số âm được biểu diễn bằng cách lấy bù 1 của số dương kể cả bit dấu.
- Ví dụ: Dùng 8 bit biểu diễn số +25 và -25 dưới dạng bù 1

  Ta biết $25_{10} = 00011001_2$

  $-25_{10} = 11100110_2$ (bù 1)

Biểu diễn số nguyên có dấu ở dạng bù 2

- Đối với số dương thì biểu diễn giống dấu và trị tuyệt đối
- Đối với số âm thì được biểu diễn dưới dạng bit dấu và giá trị của số đó ở dạng bù 2. Ta cũng có thể hiểu là số âm được biểu diễn bằng cách lấy bù 2 của số dương kể cả bit dấu.
- Ví dụ: Dùng 8 bit biểu diễn số +25 và -25 dưới dạng bù 2

  Ta biết $25_{10} = 00011001_2$

  $-25_{10} = 11100111_2$
- ***Chú ý: Số dương biểu diễn ở cả 3 cách là như nhau, chỉ khác nhau khi đó là số âm***

Phép cộng trừ nhị phân dùng bù 1

- Phép cộng giống như cộng các số nhị phân không dấu, cộng cả bit dấu.
- ***Cần lưu ý: Cộng số nhớ của bit lớn nhất vào bit cuối cùng***
- Ví dụ:

13	**0**01101	-13		**1**10010			
+	+		+		+		
11	**0**01011	-11		**1**10100			
24	**0**11000	-24	Nhớ 1	**1**00110			
			+	1			
				100111			

- Phép trừ thực hiện thông qua phép cộng

- Ví dụ:

13	**0**01101	-13	**1**10010
+	+	+	+
-11	**1**10100	11	**0**01011
2	Nhớ 1 **0**00001	-2	**1**11101
	+ 1		
	000010		

Phép cộng trừ nhị phân dùng bù 2

- Quy tắc: -A = bù 2 của A
- A – B = A + (-B) = A + (bù 2 của B)
- Ví dụ: 13 – 6 = 13 + (-6)

```
 6   =     00000110
-6   =     11111010
13   =     00001101
-----------------------
     =   100000111  (7)
```

Bỏ bit tràn (nếu có)

❑ Ví dụ: 6 – 13 = 6 + (-13)

```
13    =    00001101
-13   =    11110011
 6    =    00000110
-----------------------
      =    11111001 (-7)
```

❑ Ví dụ: - 6 – 13 = (-6) + (-13)

```
13    =    00001101
-13   =    11110011
- 6   =    11111010
-------------------
      =    11101101 (-19)
```

Cộng trừ số nhị phân nguyên

- Các ví dụ:

Ta thực hiện: 0111 chuyển thành 0111

-0101 +1011 (Số bù 2 của 0101)

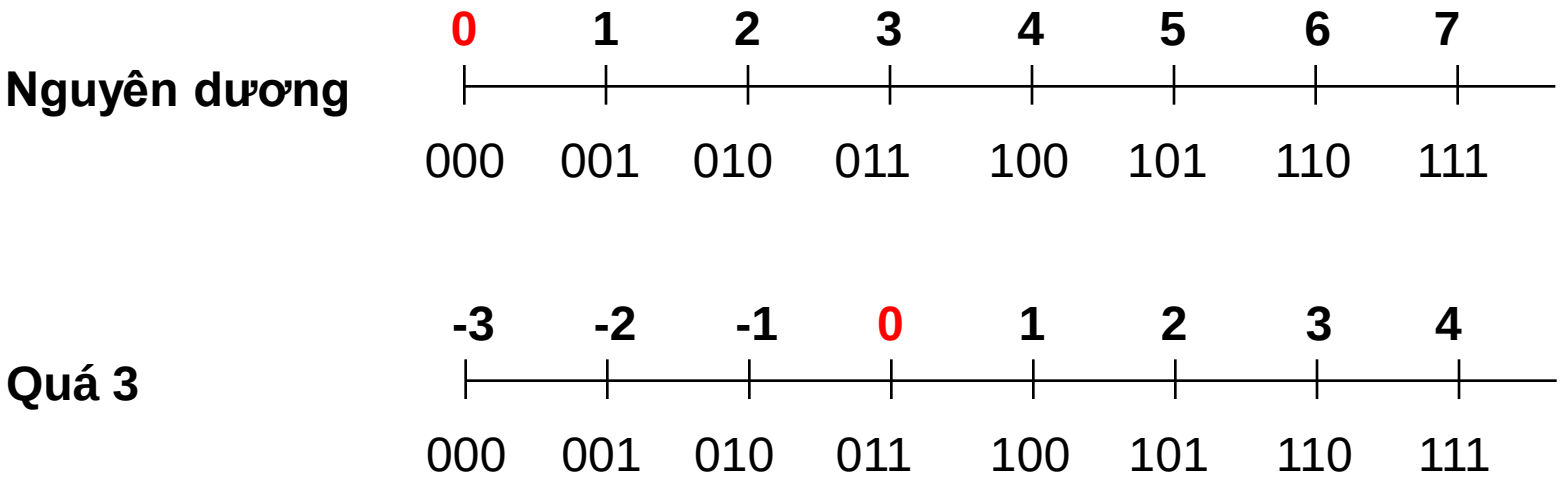
10010 Suy ra kết quả là 0010

Ta thực hiện: 0101(5) Chuyển thành 0101

-0111(-7) +1001 (Số bù 2 của 0111)

1110

Số quá n (excess-n)



Quy tắc chung:

Biểu diễn quá n của N = biểu diễn nguyên dương của (N + n)

Ví dụ:

Biểu diễn (quá 127) của 7 là:

$$127+7 = 134 = 10000110_2$$

BCD (Binary Coded Decimal)

- Biểu diễn một chữ số **thập phân** bằng 4 chữ số nhị phân (ít dùng)

```
0 = 0000    4 = 0100    8 = 1000
1 = 0001    5 = 0101    9 = 1001
2 = 0010    6 = 0110
3 = 0011    7 = 0111
```

```
  27        0010  0111
+ 36        0011  0110
----      ------------
  63        0101  1101   ──► Ký số vượt quá => kết quả sai
          +
            0000  0110   ──► Sửa sai kết quả
          ------------
            0110  0011   Kết quả = 63
```

Ví dụ tính toán với BCD

28 + 59 = 87

0010 1000
0101 1001

1000 0001 → Có nhớ 1 => kết quả sai

\+

0000 0110 → Sửa sai kết quả

1000 0111 Kết quả = 87

61 - 38 = 23

0110 0001
0011 1000

0010 1001 → Ký số bên phải mượn 1 khi trừ

\-

0000 0110 → Sửa sai kết quả

0010 0011 Kết quả = 23

Biểu diễn ký tự

- **ASCII** (7 bit) (American Standard Codes for Information Interchange) để biểu diễn 128 ký tự gọi là mã ASCII-7
- Sử dụng bộ mã ASCII mở rộng (8 bit)
  - 00 – 1F: ký tự điều khiển
  - 20 – 7F: ký tự in được
  - 80 – FF: ký tự mở rộng (ký hiệu tiền tệ, vẽ khung, …)
- Ngày nay dùng bộ mã Unicode (16 bit) (UTF-8)

Biểu diễn chấm động

- $F = (-1)^S \quad M \quad R^e$
  - S: dấu
  - M: định trị
  - R: cơ số
  - e: mũ
- Ví dụ: $2006 = (-1)^0 \quad 2.006 \quad 10^3$

Biểu diễn chấm động

- Biểu diễn chấm động được gọi là chuẩn hóa khi phần định trị chỉ có duy nhất **một** chữ số bên trái dấu chấm thập phân và chữ số đó khác không → một số chỉ có duy nhất một biểu diễn chấm động được chuẩn hóa.

2.006 10^3 (chuẩn)

20.06 10^2 (không)

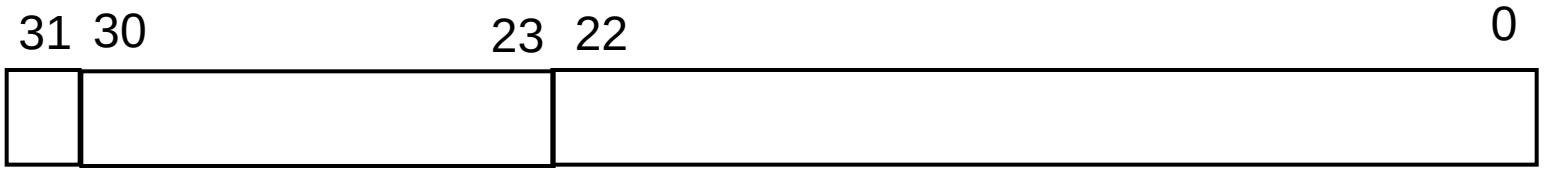
0.2006 10^4 (không)

Biểu diễn chấm động trên hệ nhị phân

- Sử dụng dạng chuẩn hóa
- Dùng 1 bit cho phần dấu: 0-dương, 1-âm
- Không biểu diễn cơ số (R) vì luôn bằng 2
- Phần định trị **chỉ biểu diễn phần lẻ** (bên phải dấu chấm) vì chữ số bên trái dấu chấm luôn là 1

Biểu diễn chấm động trên hệ nhị phân

- Ví dụ:



  - Dấu 1 bit
  - Mũ: 8 bit (từ bit 23 đến bit 30) là một số quá 127 (sẽ có trị từ -127 đến 128)
  - Định trị: 23 bit (từ bit 0 đến bit 22)

Biểu diễn chấm động trên hệ nhị phân

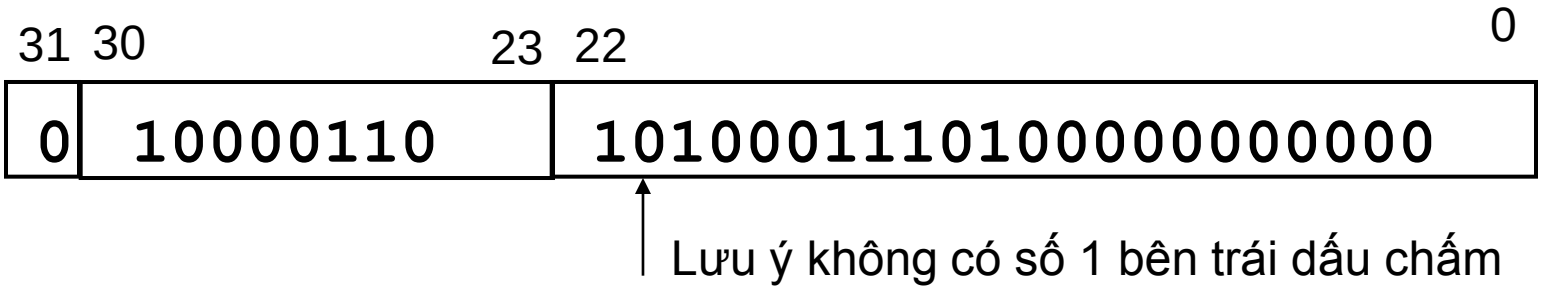
- **Ví dụ:**
- $209.8125_{10} = 11010001.1101_2$

$= 1.10100011101 \quad 2^7$

Biểu diễn (quá 127) của 7 là:

$127+7 = 134 = 10000110_2$

Kết quả:



CÂU HỎI VÀ BÀI TẬP CHƯƠNG III

Chương 4 – Mạch Logic số

4.1. Cổng và đại số Boolean

Cổng – cơ sở phần cứng, từ đó chế tạo ra mọi máy tính số

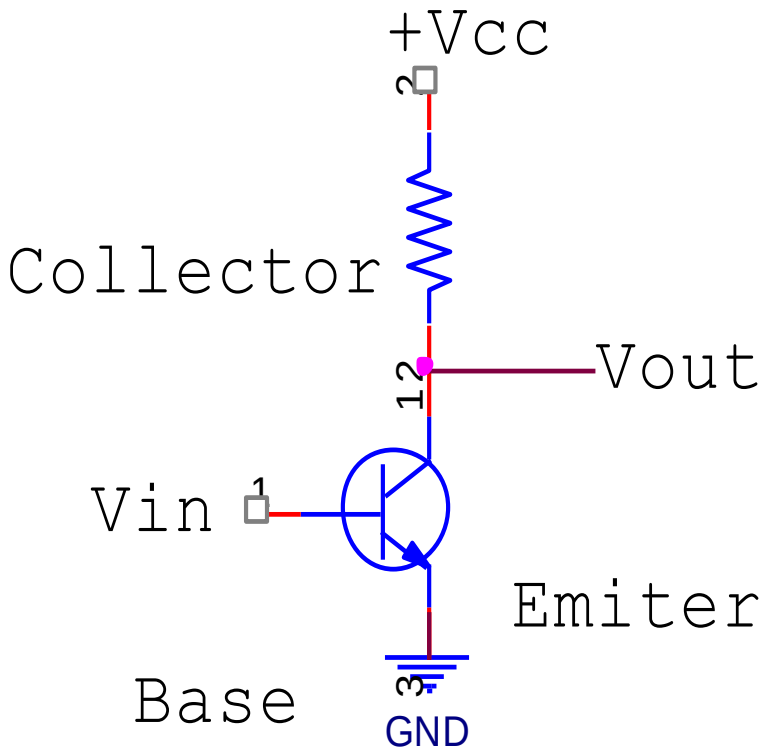
Gọi là cổng luận lý vì nó cho kết quả lý luận của đại số logic như nếu A đúng và B đúng thì C đúng (cổng A AND B = C)

Mạch số là mạch trong đó chỉ hiện diện hai giá trị logic. Thường tín hiệu giữa 0 và 1 volt đại diện cho số nhị phân 0 và tín hiệu giữa 2 và 5 volt – nhị phân 1.

4.1.1. Cổng (Gate)

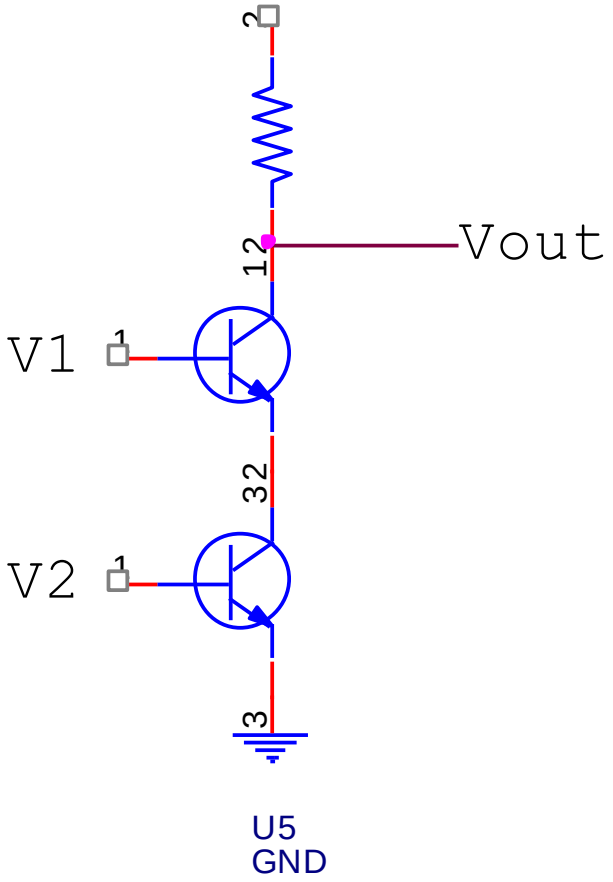
- Bộ chuyển đổi transistor – cổng (gate): Cực góp (collector), cực nền (base), cực phát (emitter)

a) Cổng INV (NOT)



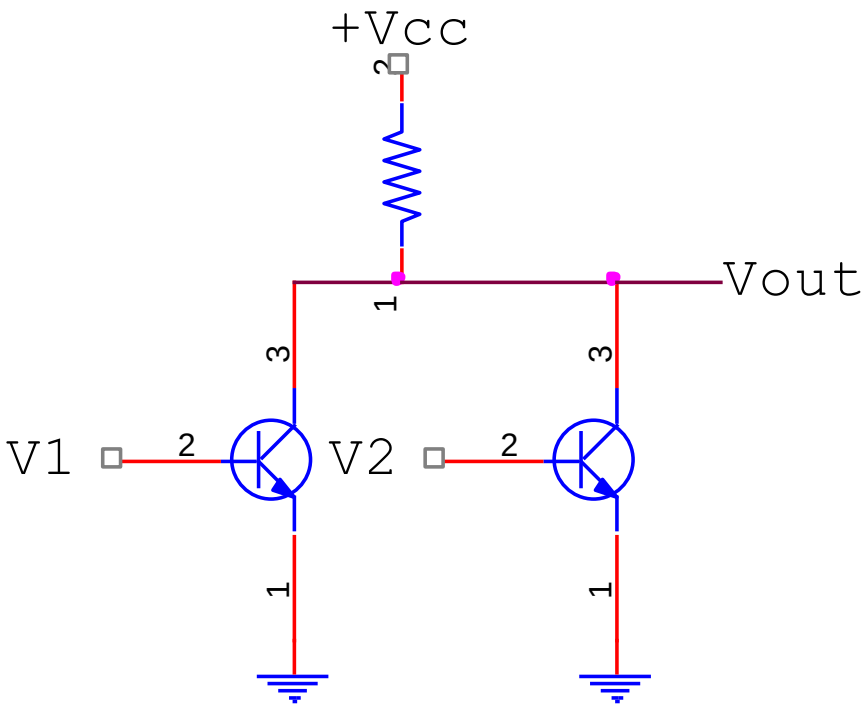
Cổng NAND

b)



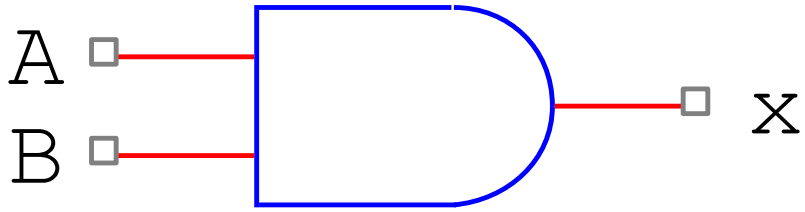
4.1.1. Cổng (Gate)

- Cổng NOR



Các cổng cơ bản của logic số

- AND
- OR
- Inverter
- NAND
- NOR
- XOR (exclusive-OR)
- NXOR

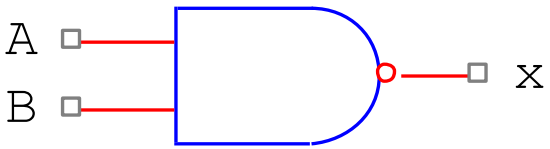


A	B	x
0	0	0
0	1	0
1	0	0
1	1	1

AND

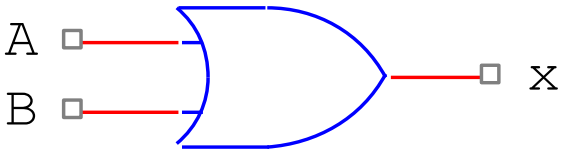
Các cổng cơ bản của logic số

NAND



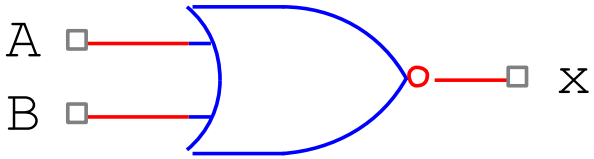
A	B	x
0	0	1
0	1	1
1	0	1
1	1	0

OR



A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

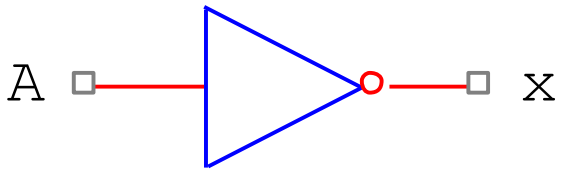
NOR

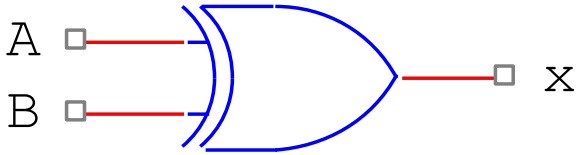


A	B	x
0	0	1
0	1	0
1	0	0
1	1	0

Các cổng cơ bản của logic số

- Cổng INVERTER (NOT) và cổng XOR





A	x
0	1
1	0

A	***B***	***f***
0	0	0
0	1	1
1	0	1
1	1	0

4.1.2. Đại số Boolean (Boolean Algebra)

- Đại số Boolean được lấy theo tên người khám phá ra nó, nhà toán học người Anh George Boole.

- Đại số Boolean là môn đại số trong đó biến và hàm chỉ có thể lấy giá trị 0 và 1.

-Đại số boolean còn gọi là đại số chuyển mạch (switching algebra)

Logic 0	Logic 1
Sai	Đúng
Tắt	Mở
Thấp	Cao
Không	Có
Công tắc mở	Công tắc đóng

4.1.2. Đại số Boolean (Boolean Algebra)

Tên	Dạng AND	Dạng OR
Định luật thống nhất	1A = A	0 + A = A
Định luật không	0A = 0	1+ A = 1
Định luật Idempotent	AA = A	A + A = A
Định luật nghịch đảo	$A\overline{A}$ [redacted] 0	A [redacted] $\overline{A}$ [redacted] 1
Định luật giao hoán	AB = BA	A + B = B + A
Định luật kết hợp	(AB)C = A(BC)	(A+B)+C = A + (B+C)
Định luật phân bố	A + BC = (A + B)(A + C)	A(B+C) = AB + AC
Định luật hấp thụ	A(A + B) = A	A + AB = A
Định luật De Morgan	$\overline{AB}$ [redacted] $\overline{A}$ [redacted] $\overline{B}$	$\overline{A}$ [redacted] $\overline{B}$ [redacted] $\overline{AB}$

4.1.2. Đại số Boolean (Boolean Algebra)

- Quy tắc về phủ định:

$$\overline{\overline{X}} = X$$

- ***Hàm Logic:*** $y = A \; OR \; B$

$$= A + B$$

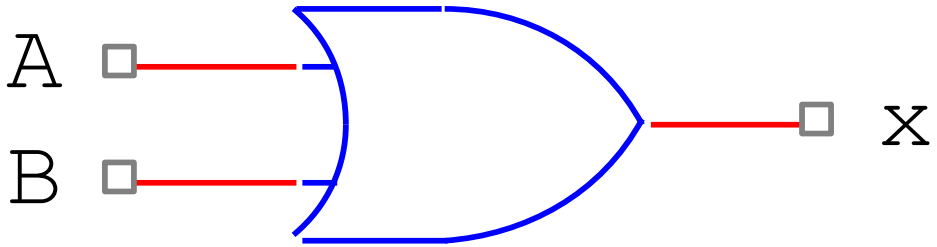
- ***Bảng chân trị*** (truth table)

A	***B***	***y***
0	0	0
0	1	1
1	0	1
1	1	1

Phép toán OR và cổng OR

- ***Bảng chân trị*** (truth table), ký hiệu phép toán, ký hiệu cổng

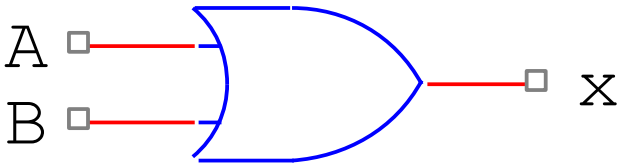
A	*B*	*x=A+B*
0	0	0
0	1	1
1	0	1
1	1	1

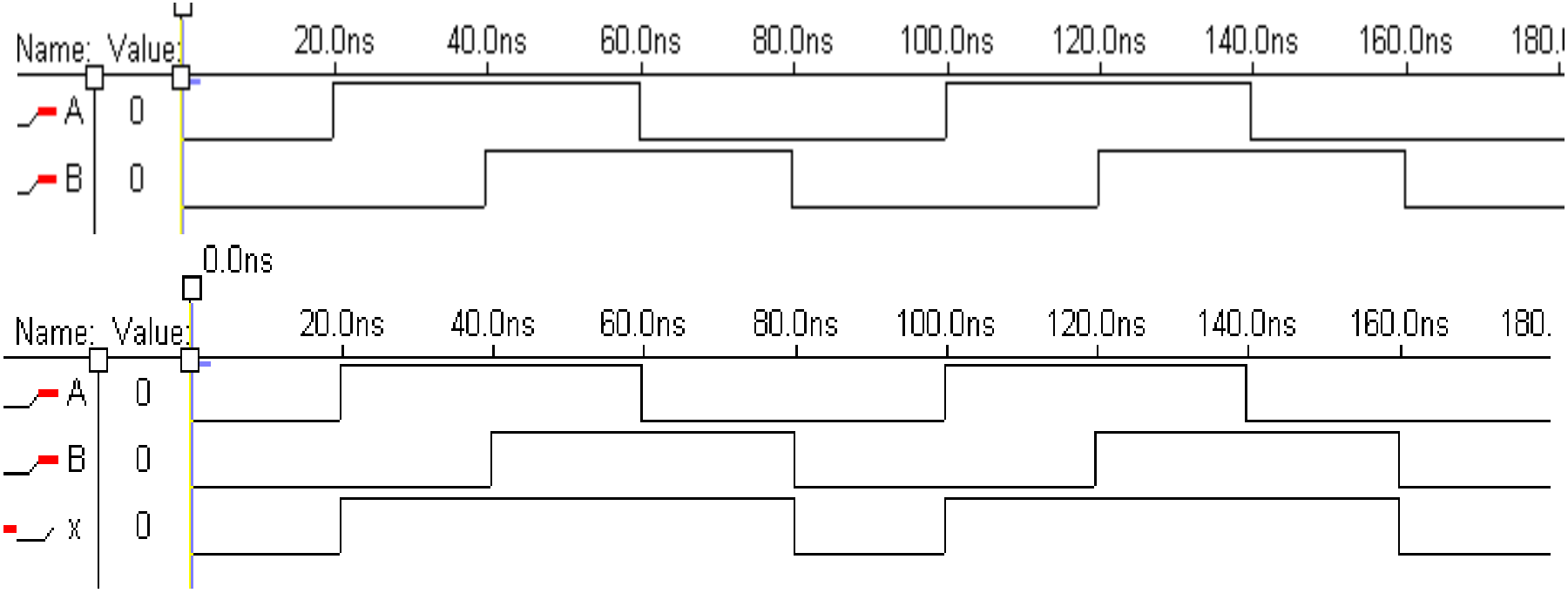


- Phép toán cho 3 biến, 4 biến,…
- Phép toán AND, NOT, XOR

Phép toán OR và cổng OR

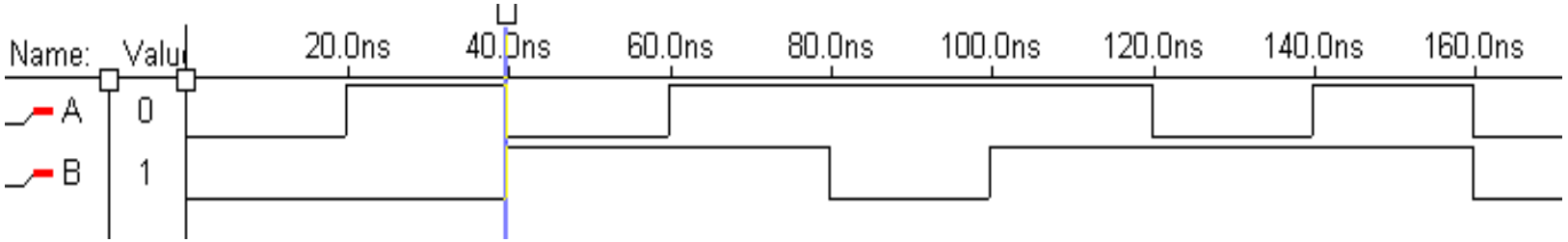
- Biểu đồ (Sơ đồ) thời gian. VD:





4.1.2. Đại số Boolean (Boolean Algebra)

- Phép toán AND với cổng AND
- Phép toán INVerter (NOT) với cổng NOT
- Phép toán XOR với cổng XOR
- Ví dụ:
  - Xác định đầu ra x từ cổng AND, nếu các tín hiệu đầu vào có dạng hình 4.4:

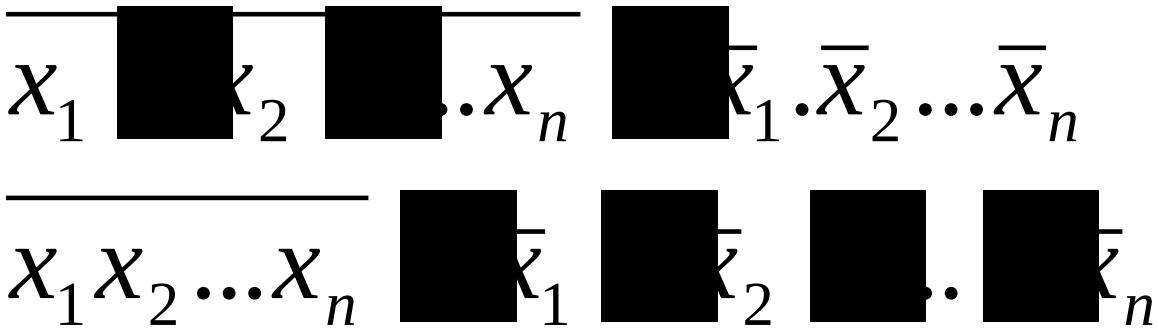


Hàm của n biến logic sẽ có 2^n tổ hợp biến,

4.1.2. Đại số Boolean (Boolean Algebra)

- **Định lý DeMorgan**

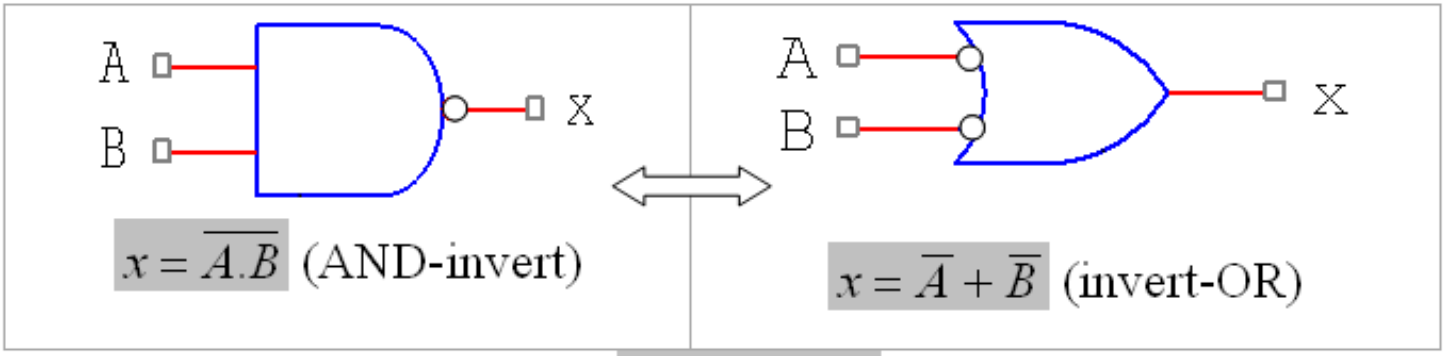
- **Dạng tổng quát:**



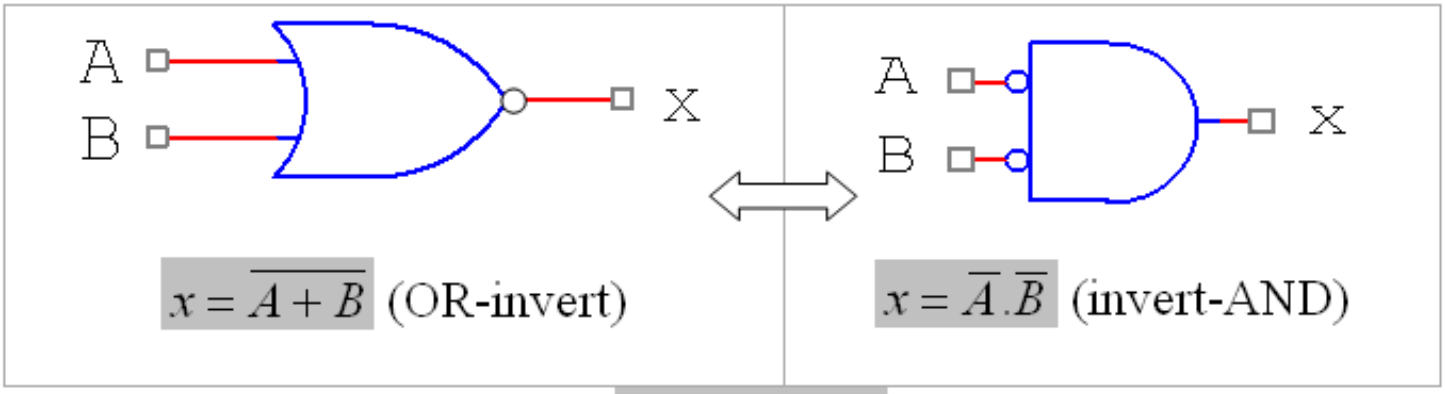
- **Ví dụ:**

4.1.2. Đại số Boolean (Boolean Algebra)

- Các cổng tương đương từ **định lý DeMorgan**



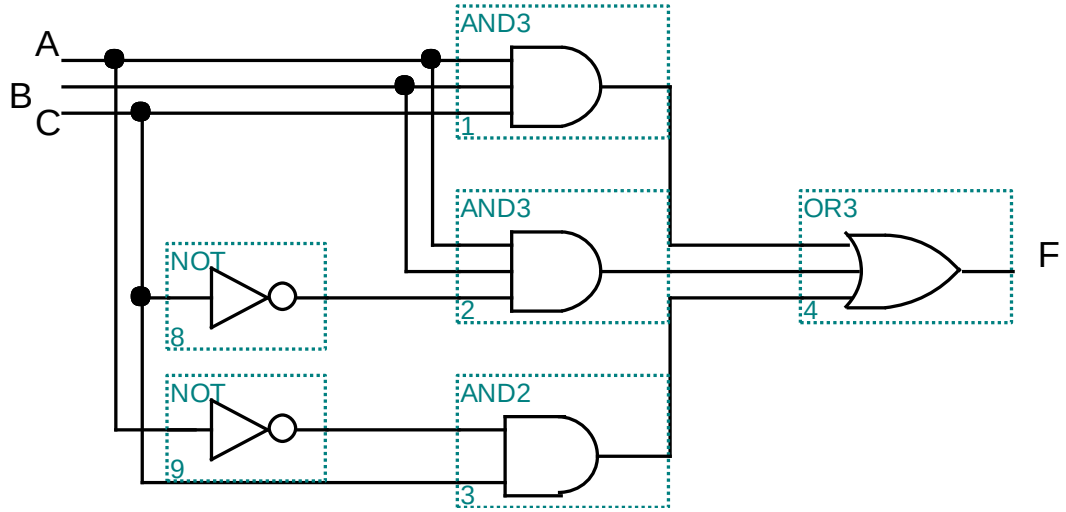
a) $\overline{AB} = \overline{A} + \overline{B}$



b) $\overline{A + B} = \overline{A}.\overline{B}$

4.1.2. Đại số Boolean (Boolean Algebra)

- Một số ví dụ:
  - Đơn giản hàm Boolean
  - Đơn giản mạch
  - Thiết kế mạch



$F = ABC + AB\bar{C} + \bar{A}C$

Đơn giản???

4.1.2. Đại số Boolean (Boolean Algebra)

- ***Ví dụ 1:***

  Dùng bảng chân trị để biểu diễn hàm *f = (A AND B) OR (C AND NOT B)*, vẽ sơ đồ mạch cho hàm *f*.

- ***Ví dụ 2:***

Dùng Boolean Algebra đơn giản các biểu thức sau:

a) y = A + AB

b) y = A $\overline{B}$ D + A $\overline{B}$ $\overline{D}$

c) x = $(\overline{A}$ [redacted]$B)(A$ [redacted]$B)$

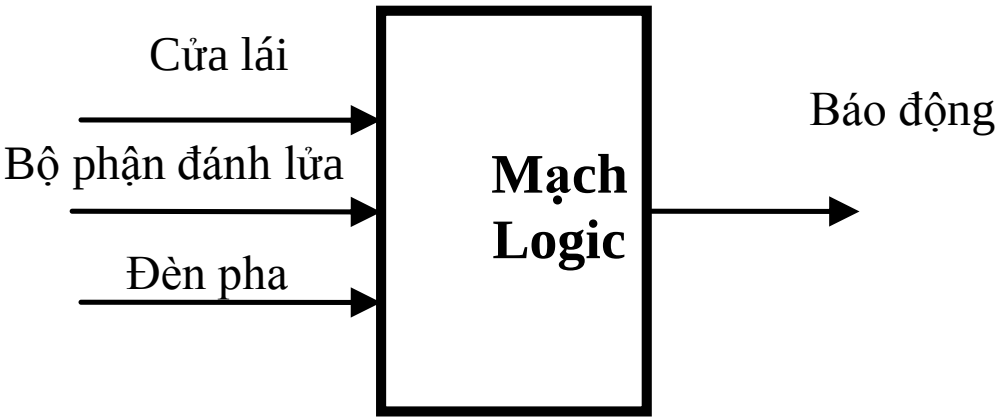
d) z [redacted]$(B\overline{C}$ [redacted]$\overline{A}D)(A\overline{B}$ [redacted]$C\overline{D})$

A	B	C	NOTB	A AND B	C AND NOT B	F
0	0	0	1	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	1	0	1	0	1

4.1.2. Đại số Boolean (Boolean Algebra)

❑ Ví dụ 3:

Để làm một bộ báo hiệu cho lái xe biết một số điều kiện, người ta thiết kế 1 mạch báo động như sau:



Tín hiệu từ :
Cửa lái: 1- cửa mở, 0 – cửa đóng;

Bộ phận đánh lửa: 1 - bật, 0 - tắt;

Đèn pha: 1 - bật, 0 - tắt.

4.2. Bản đồ Karnaugh

Khái niệm:

- Ô kế cận
- Các vòng gom chung
- Ô không xác định hay tùy định

f(A,B,C) = [redacted],2,4,5,6)

khi gom 2^n Ô kế cận sẽ loại được n biến. Những biến bị loại là những biến khi ta đi vòng qua các ô kế cận mà giá trị của chúng thay đổi.

A \ B	0	1
0	0	1
1	2	3

a) Bản đồ 2 biến

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

b) Bản đồ 3 biến

4.2. Bản đồ Karnaugh

[redacted]

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

c) Bản đồ 4 biến

[redacted]

4.2. Bản đồ Karnaugh

❑ Những điều cần lưu ý:

- Vòng gom được gọi là hợp lệ
- Biểu diễn hàm Boolean theo dạng
  - tổng các tích (dạng 1)
  - tích các tổng (dạng 2)
- Các vòng phải được gom sao cho số ô có thể vào trong vòng là lớn nhất và nhớ là để đạt được điều đó, thường ta phải gom cả những ô đã gom vào trong các vòng khác

❑ Mục đích cần đạt:

- Biểu thức có chứa ít nhất các thừa số và mỗi thừa số chứa ít nhất các biến.
- Mạch logic thực hiện có chứa ít nhất các vi mạch số.

Dạng chính tắc và dạng chuẩn của hàm Boole

- Tích chuẩn (minterm): m_i ($0 \le i \le 2^n-1$) là các số hạng tích (AND) của n biến mà hàm Boole phụ thuộc với quy ước biến đó có bù nếu nó là 0 và không bù nếu là 1.
- Tổng chuẩn (Maxterm): M_i ($0 \le i \le 2^n-1$) là các số hạng tổng (OR) của n biến mà hàm Boole phụ thuộc với quy ước biến đó có bù nếu nó là 1 và không bù nếu là 0

x	y	z	Minterms	Maxterms
0	0	0	$m_0 = \bar{x}\,\bar{y}\,\bar{z}$	$M_0 = x + y + z$
0	0	1	$m_1 = \bar{x}\,\bar{y}\,z$	$M_1 = x + y + \bar{z}$
0	1	0	$m_2 = \bar{x}\,y\,\bar{z}$	$M_2 = x + \bar{y} + z$
0	1	1	$m_3 = \bar{x}\,y\,z$	$M_3 = x + \bar{y} + \bar{z}$
1	0	0	$m_4 = x\,\bar{y}\,\bar{z}$	$M_4 = \bar{x} + y + z$
1	0	1	$m_5 = x\,\bar{y}\,z$	$M_5 = \bar{x} + y + \bar{z}$
1	1	0	$m_6 = x\,y\,\bar{z}$	$M_6 = \bar{x} + \bar{y} + z$
1	1	1	$m_7 = x\,y\,z$	$M_7 = \bar{x} + \bar{y} + \bar{z}$

Dạng chính tắc (Canonical Form)

- **Dạng chính tắc 1:** là dạng tổng của các tích chuẩn_1 (minterm-_1 là minterm mà tại tổ hợp đó hàm Boole có giá trị 1).
-

x	y	z	Minterms	Maxterms	F	$\overline{F}$
0	0	0	$m_0 = \overline{x}\,\overline{y}\,\overline{z}$	$M_0 = x + y + z$	0	1
0	0	1	$m_1 = \overline{x}\,\overline{y}\,z$	$M_1 = x + y + \overline{z}$	1	0
0	1	0	$m_2 = \overline{x}\,y\,\overline{z}$	$M_2 = x + \overline{y} + z$	0	1
0	1	1	$m_3 = \overline{x}\,y\,z$	$M_3 = x + \overline{y} + \overline{z}$	1	0
1	0	0	$m_4 = x\,\overline{y}\,\overline{z}$	$M_4 = \overline{x} + y + z$	1	0
1	0	1	$m_5 = x\,\overline{y}\,z$	$M_5 = \overline{x} + y + \overline{z}$	0	1
1	1	0	$m_6 = x\,y\,\overline{z}$	$M_6 = \overline{x} + \overline{y} + z$	0	1
1	1	1	$m_7 = x\,y\,z$	$M_7 = \overline{x} + \overline{y} + \overline{z}$	0	1

$$F(x, y, z) = \overline{x}\,\overline{y}\,z + \overline{x}\,y\,z + x\,\overline{y}\,\overline{z} = m_1 + m_3 + m_4$$
$$= \sum(1,3,4)$$

Dạng chính tắc (Canonical Form) (tt)

- **Dạng chính tắc 2:** là dạng tích của các tổng chuẩn_0 (Maxterm-_0 là Maxterm mà tại tổ hợp đó hàm Boole có giá trị 0).

$$F(x, y, z) = [redacted]$$

$$[redacted] = [redacted]\,M_5 M_6 M_7$$

$$F(x, y, z) = [redacted]\,M_5 M_6 M_7$$

$$= [redacted]$$

- **Trường hợp tùy định (don't care)**

Hàm Boole theo dạng chính tắc:

$$F(A, B, C) = [redacted]2, 3, 5) + d(0, 7)$$

$$= [redacted]1, 4, 6) \cdot D(0, 7)$$

A	B	C	F
0	0	0	X
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	X

Dạng chuẩn (Standard Form)

- **Dạng chuẩn 1:** là dạng tổng các tích (S.O.P – Sum of Product)

  Vd: $F(x, y, z) = xy + z$

  Ta có thể chuyển về dạng chính tắc 1 bằng cách thêm vào các cặp không phụ thuộc dạng $(x+\overline{x})$ hoặc dạng chính tắc 2 bằng $x.\overline{x}$

- **Dạng chuẩn 2:** là dạng tích các tổng (P.O.S –Product of Sum)

Vd: $F(x, y, z) = (x + \overline{z})\,\overline{y}$

Ta có thể chuyển về dạng chính tắc 1 hoặc dạng chính tắc 2

4.2. Bản đồ Karnaugh

- ***Ví dụ 1:***

  Dùng bản đồ Karnaugh đơn giản hàm f(A,B,C) = [redacted]2,4,5,6)

- ***Ví dụ 2:***

  Dùng bản đồ Karnaugh rút gọn hàm

  $f(A,B,C,D)$ [redacted]

  và vẽ sơ đồ mạch của hàm f dùng các cổng *AND*, *OR* và *NOT*.

- ***Ví dụ 3:***

  $f(A,B,C,D)$ [redacted]$,7,8,9,10,11,13)$

- ***Ví dụ 4:***

  Cực tiểu các hàm trên ở dạng tích các tổng

4.3. Những mạch logic số cơ bản

Mạch tích hợp IC (Intergrated Circuit)

Mạch kết hợp (Combinational circuit)

Mạch Giải Mã & Mã Hóa

Mạch Tuần Tự

Mạch Tích hợp *IC (Intergrated Circuit)*

Mạch Tích hợp

Các linh kiện điện tử được gắn trên cùng một bản mạch và nối với nhau thông qua các đường khắc dẫn tín hiệu trên bản mạch này. Các mạch này ngày càng thu nhỏ lại gọi là mạch tích hợp – Integrated circuit (IC)

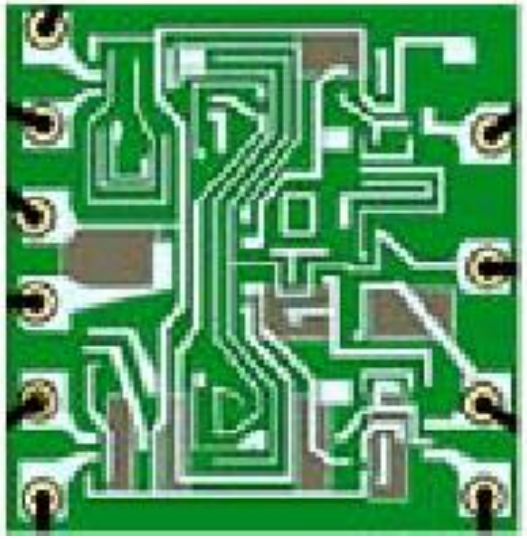
IC được chia thành các loại dưới đây tùy thuộc vào khả năng chứa và sắp xếp các cổng trên cùng một chip gọi là mức tích hợp:

- Mạch SSI (cỡ nhỏ): 1-10 cổng
- Mạch MSI (trung bình): 10-100 cổng
- Mạch LSI (cỡ lớn): 100-100.000 cổng
- Mạch VLSI (rất lớn): > 100.000 cổng

Một số vi mạch SSI

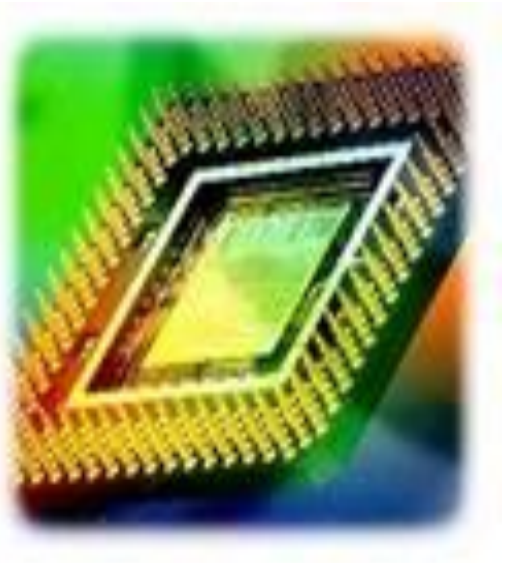
CHIP

Các IC được nén lại và đóng gói vào trong 1 vỏ bọc bằng gốm (Ceramic), hoặc chất dẻo có các chân ra ngoài gọi là CHIP.



Các kiểu đóng gói CHIP

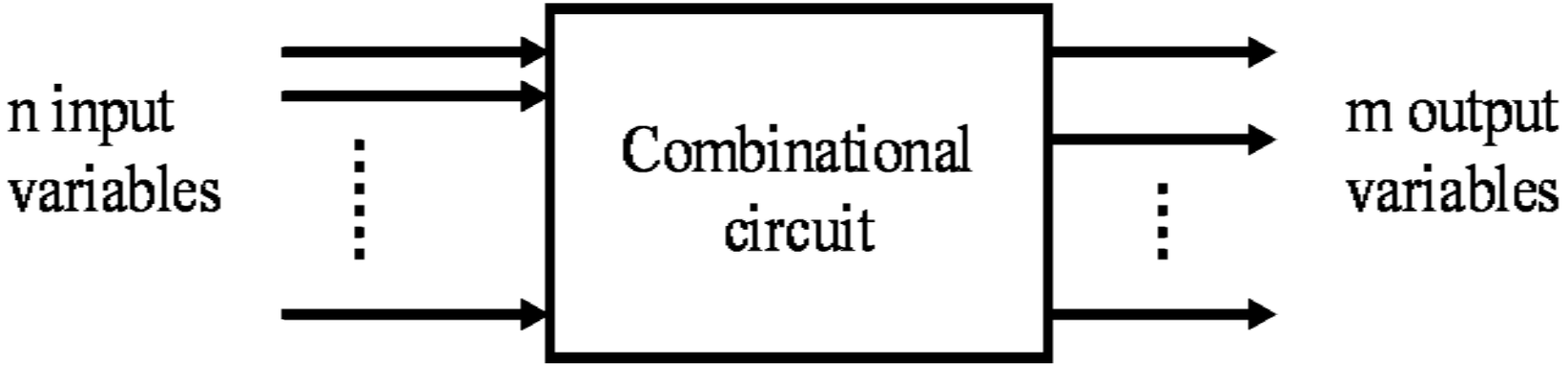
- Dual Inline Package (DIP)
- Pin Grid Array (PGA)
- Plastic Quad Flat Pack



Mạch kết hợp (tổ hợp) (Combinational circuit)

1. Định nghĩa

Mạch kết hợp là tổ hợp các cổng luận lý kết nối với nhau tạo thành một bản mạch có chung một tập các ngõ vào và ra.



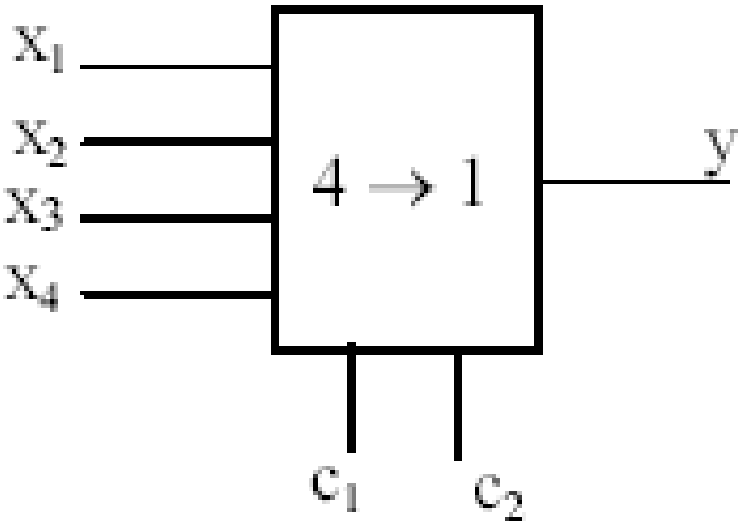
Lược đồ khối mạch kết hợp

2. Các bước thiết kế mạch kết hợp

1. Xác định bài toán để đi đến kết luận có những đầu nhập, xuất nào
2. Lập bảng chân trị xác định mối quan hệ giữa nhập và xuất
3. Dựa vào bảng chân trị, xác định hàm cho từng ngõ ra
4. Dùng đại số boolean hoặc bản đồ Karnaugh để đơn giản các hàm ngõ ra
5. Vẽ sơ đồ mạch theo các hàm đã đơn giản.

Bộ dồn kênh (Multiplexer)

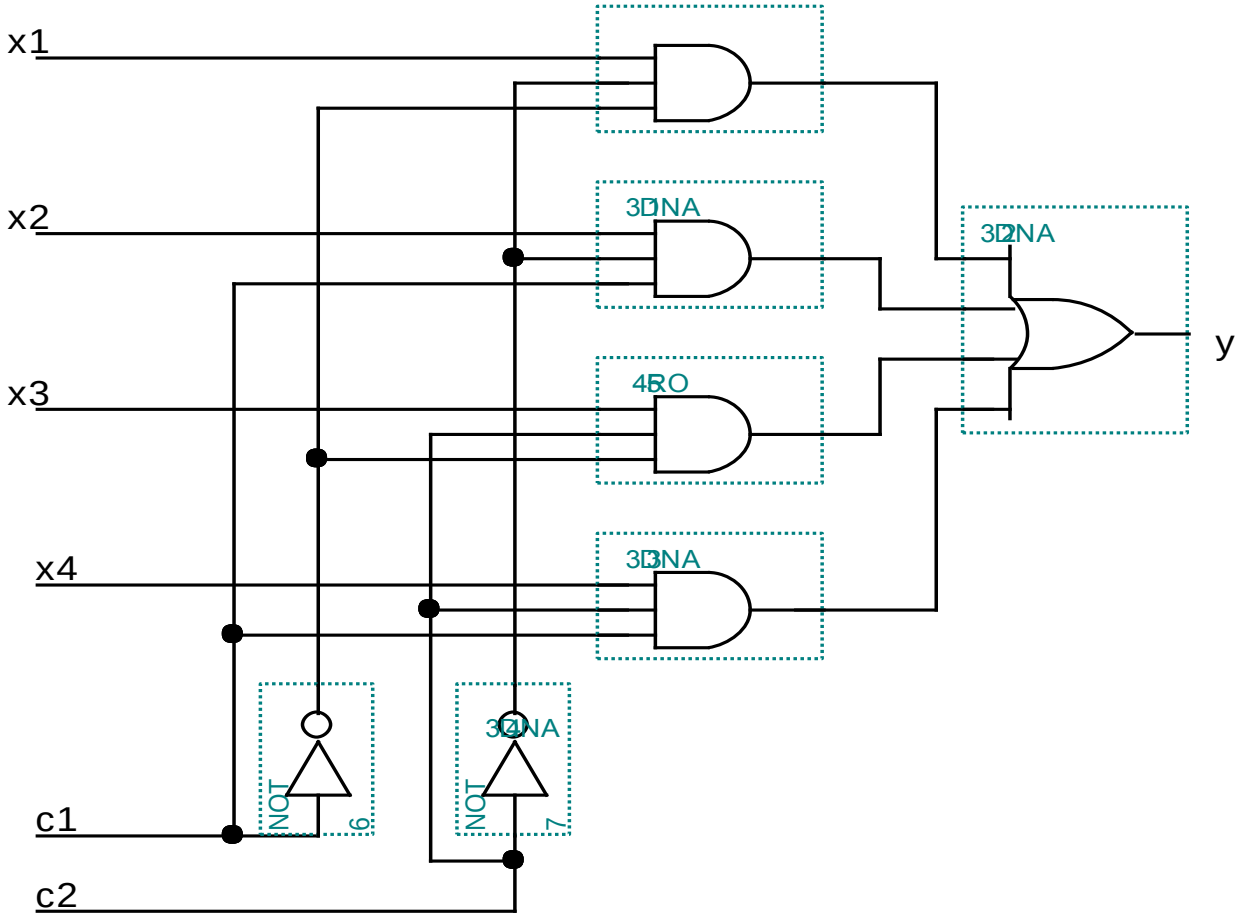
- Bộ dồn kênh hay còn gọi là mạch chọn kênh là mạch có chức năng chọn lần lượt 1 trong N kênh vào để đưa đến ngõ ra duy nhất



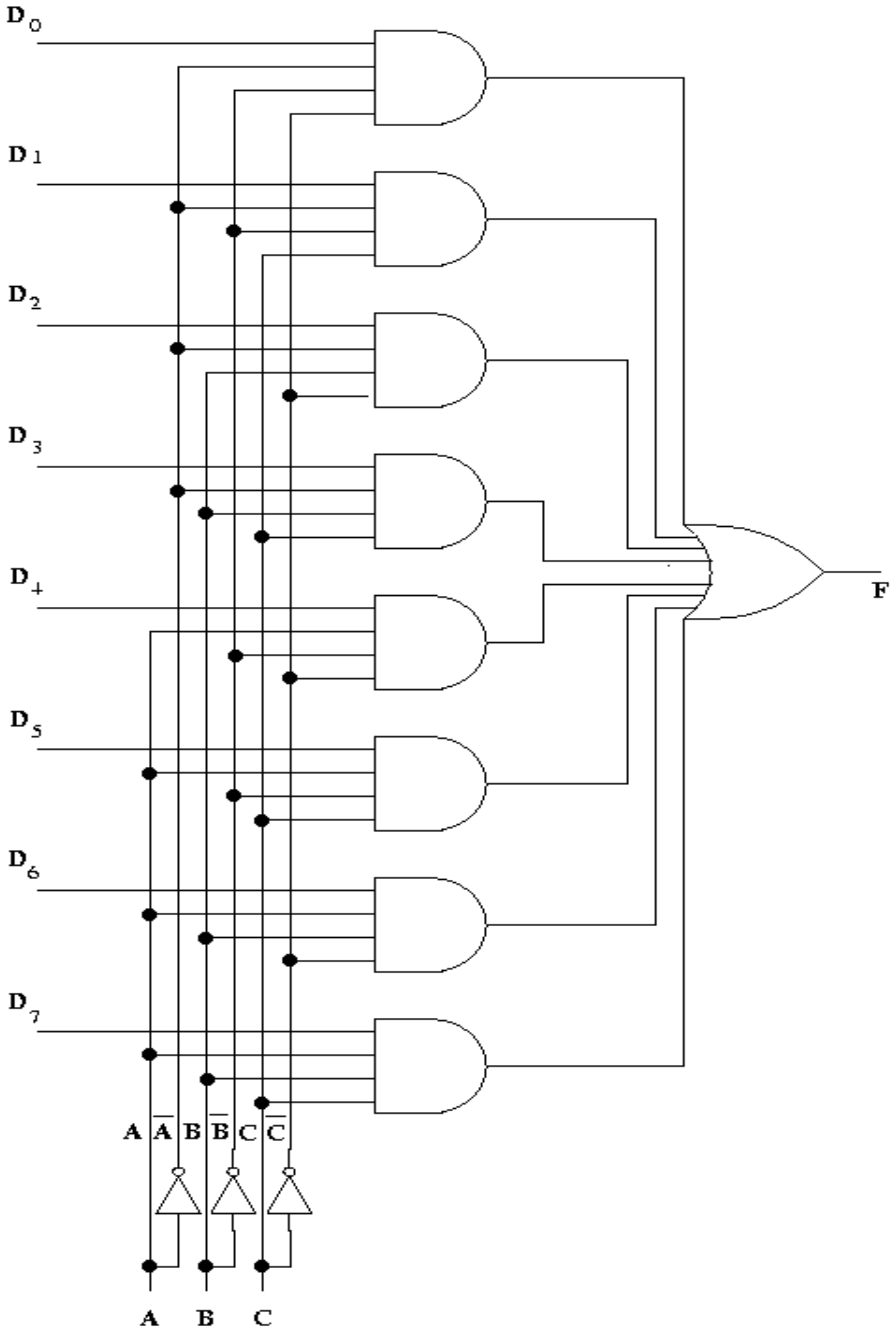
c2	*c1*	*y*
0	0	x_1
0	1	x_2
1	0	x_3
1	1	x_4

Bộ dồn kênh (Multiplexer)

- Sơ đồ bộ dồn kênh 4 đầu vào, 1 đầu ra

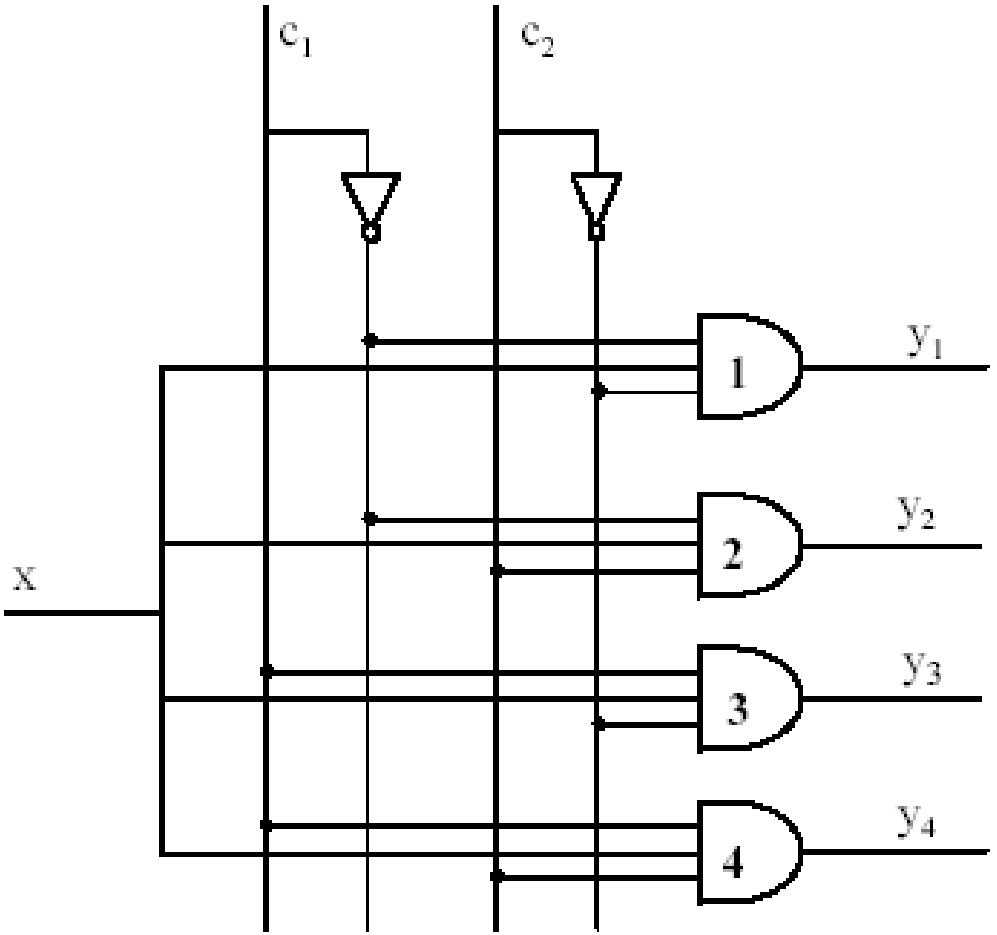


Bộ dồn kênh (Multiplexer) 8 đầu vào



Bộ phân kênh (*Demultiplexer*)

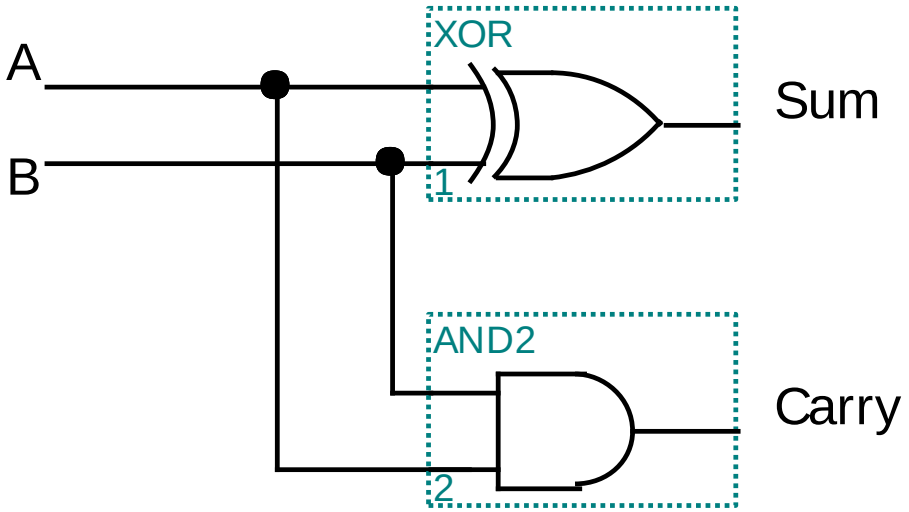
c_1	c_2	y_1	y_2	y_3	y_4
0	0	**x**	0	0	0
0	1	0	**x**	0	0
1	0	0	0	**x**	0
1	1	0	0	0	**x**



Mạch cộng (adder)

bộ nửa cộng (half adder)

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

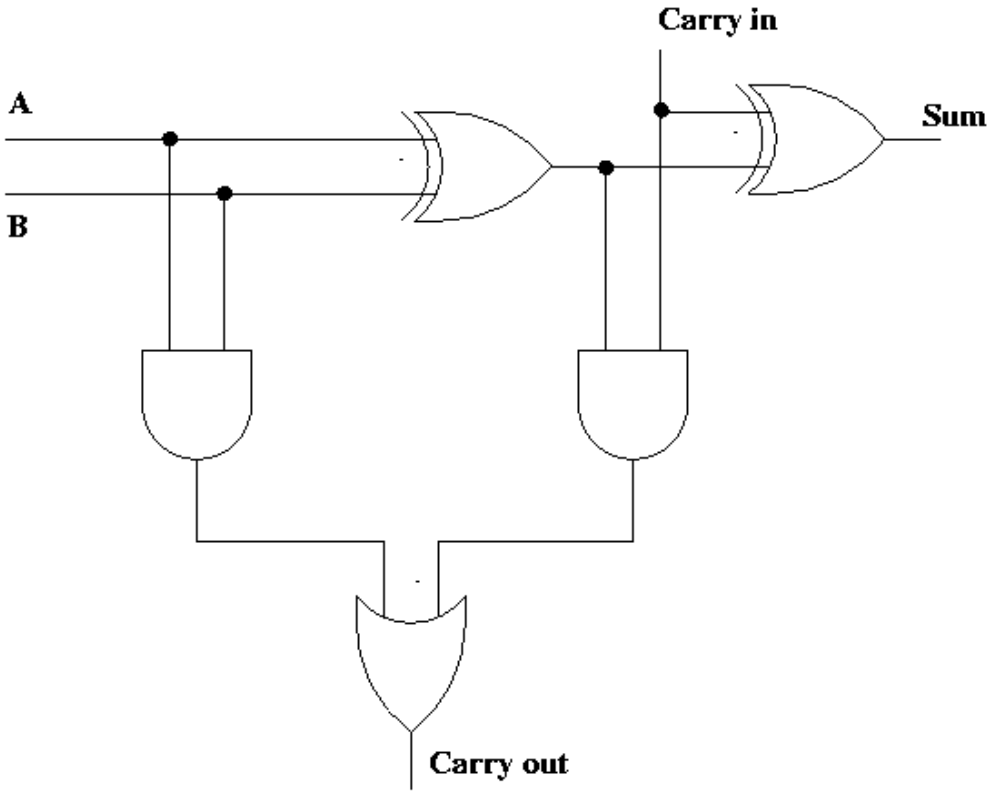


Bảng chân trị và mạch cho bộ nửa cộng

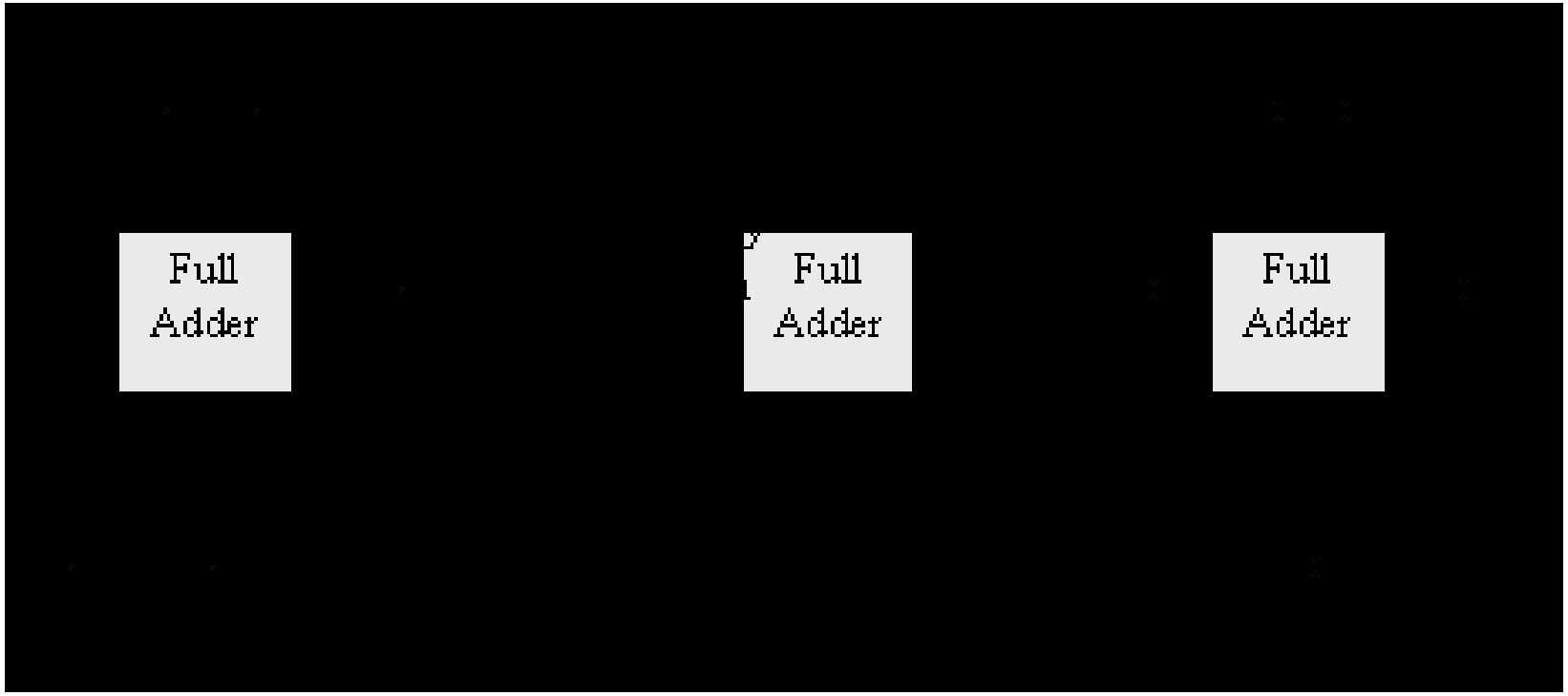
Mạch cộng (adder)

❑ Bộ cộng đầy đủ(*Full Adder*)

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



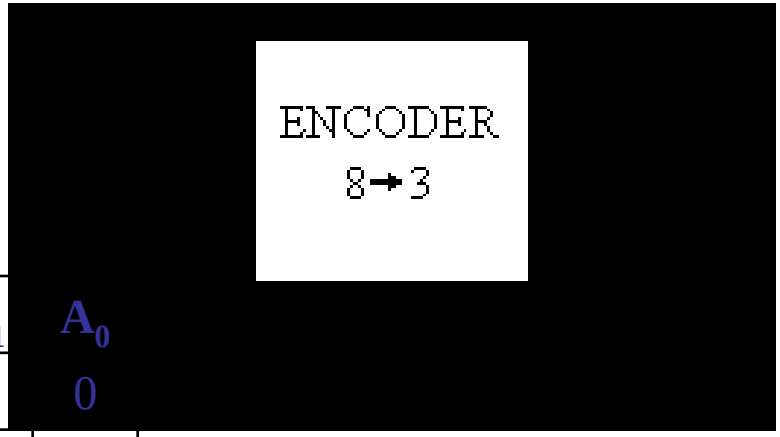
Bộ cộng n bit



Mạch giải mã và mã hóa

❑ *Mạch mã hoá (Encoder)*

2^n ngõ nhập → n ngõ xuất

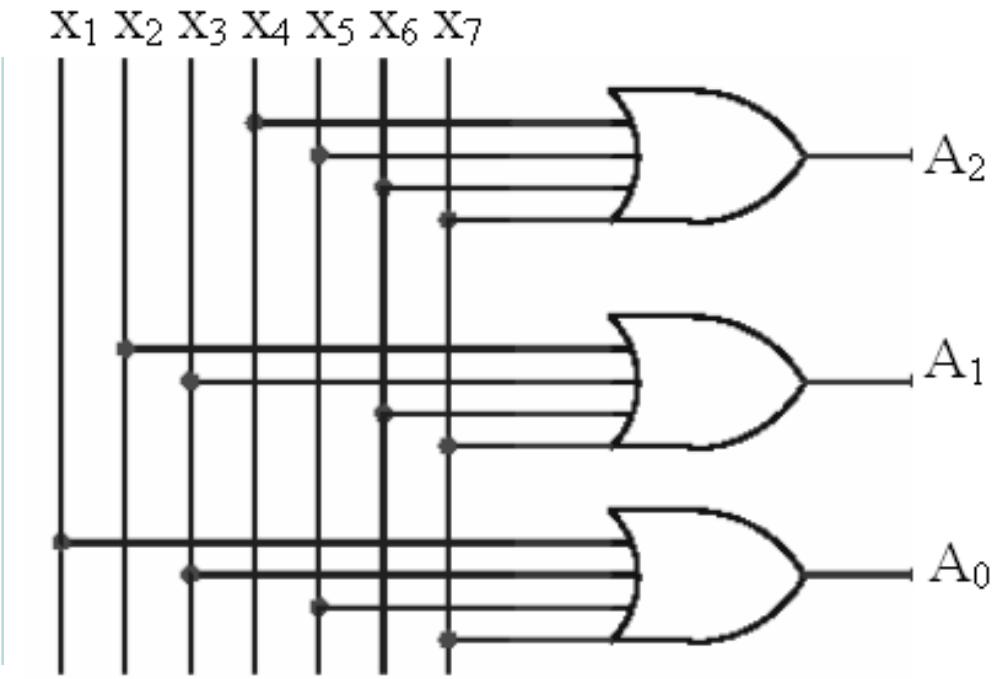


x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Mạch giải mã và mã hóa

- Phương trình logic tối giản:
- A0 = x1 + x3 + x5 + x7
- A1 = x2 + x3 + x6 + x7
- A2 = x4 + x5 + x6 + x7

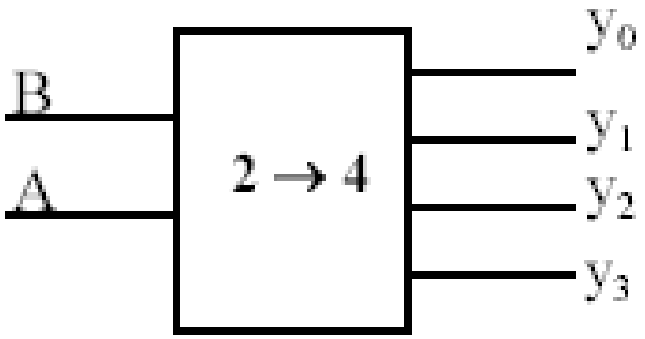
ENCODER 8→3



Mạch giải mã (Decoder)

n ngõ nhập → 2^n ngõ xuất

B	A	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



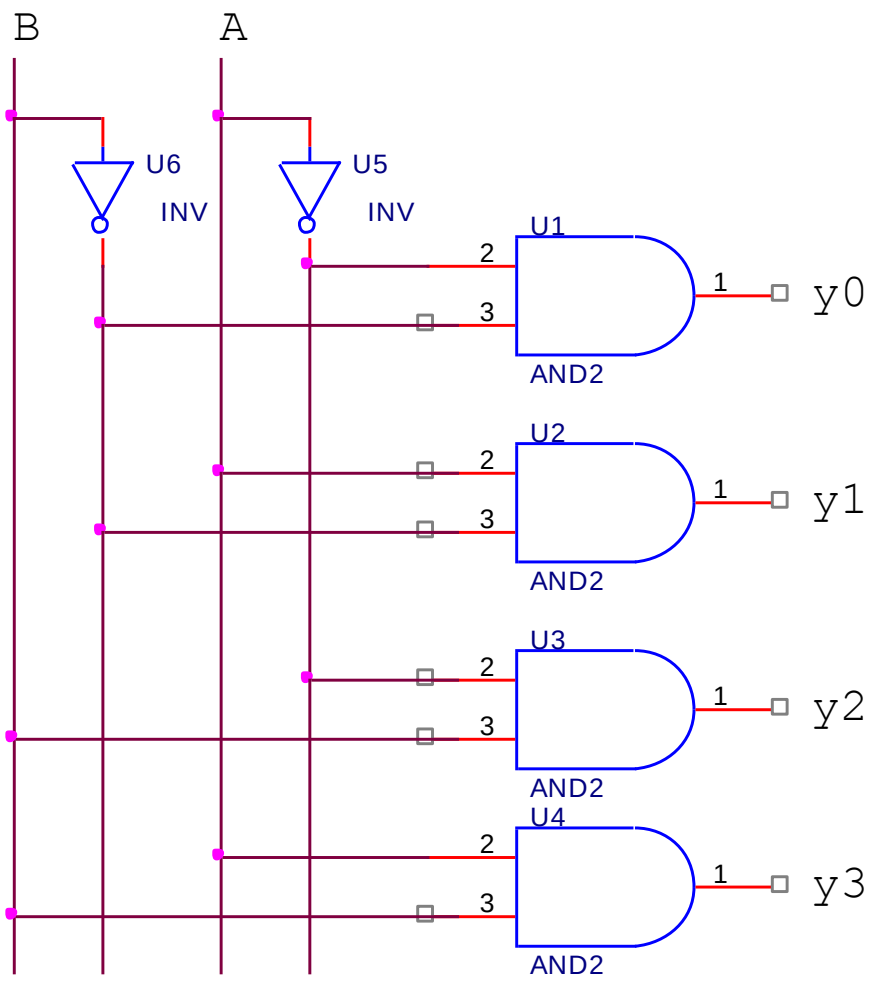
Nếu ngõ nhập có một số tổ hợp không dùng thì số ngõ ra có thể ít hơn 2^n .

Khi đó mạch giải mã gọi là mạch giải mã n-m, với m [redacted]2^n

Mạch giải mã (Decoder)

- phương trình logic tối giản

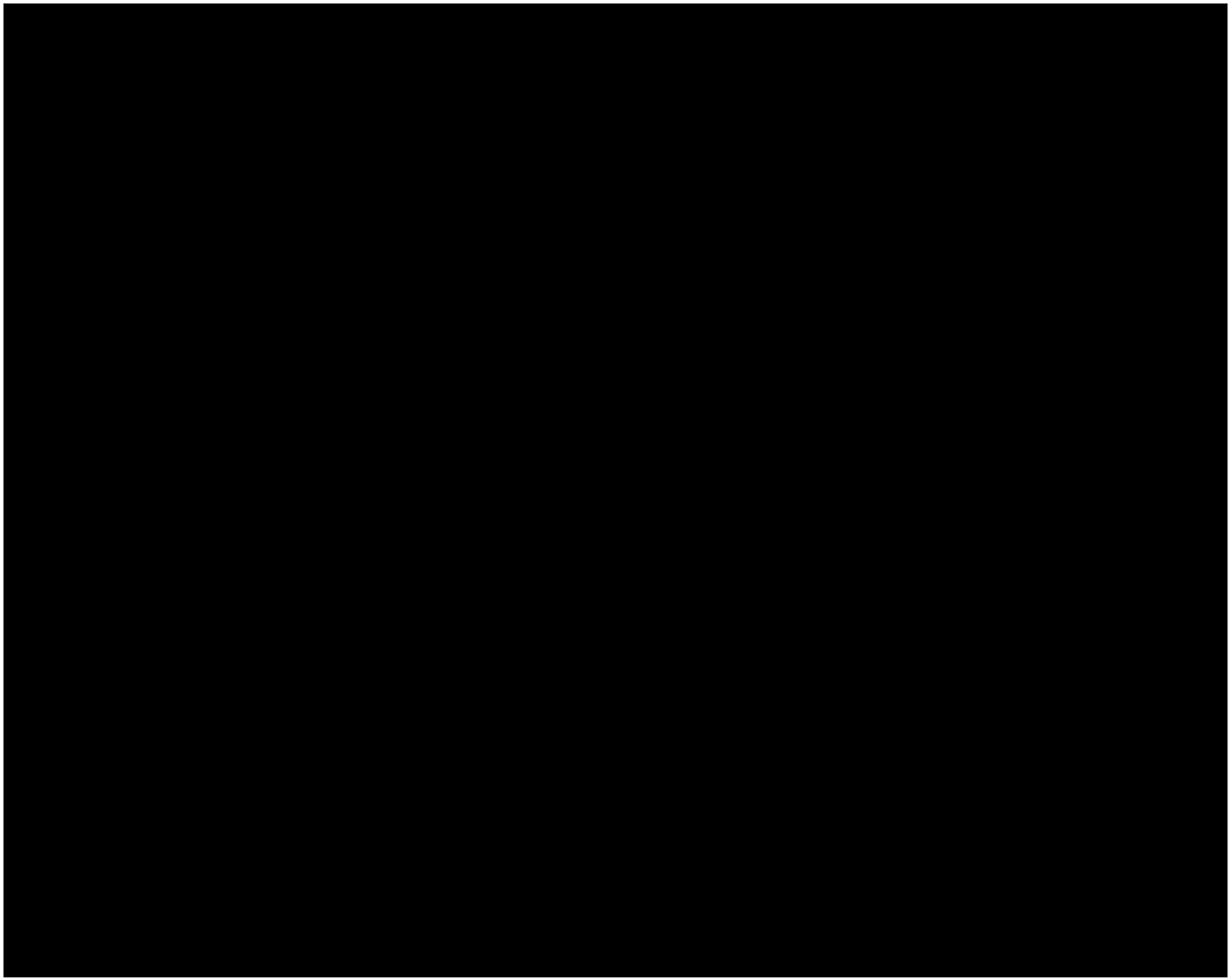
$$y_0 = \overline{A}\,\overline{B}$$
$$y_1 = A\overline{B}$$
$$y_2 = \overline{A}B$$
$$y_3 = AB$$



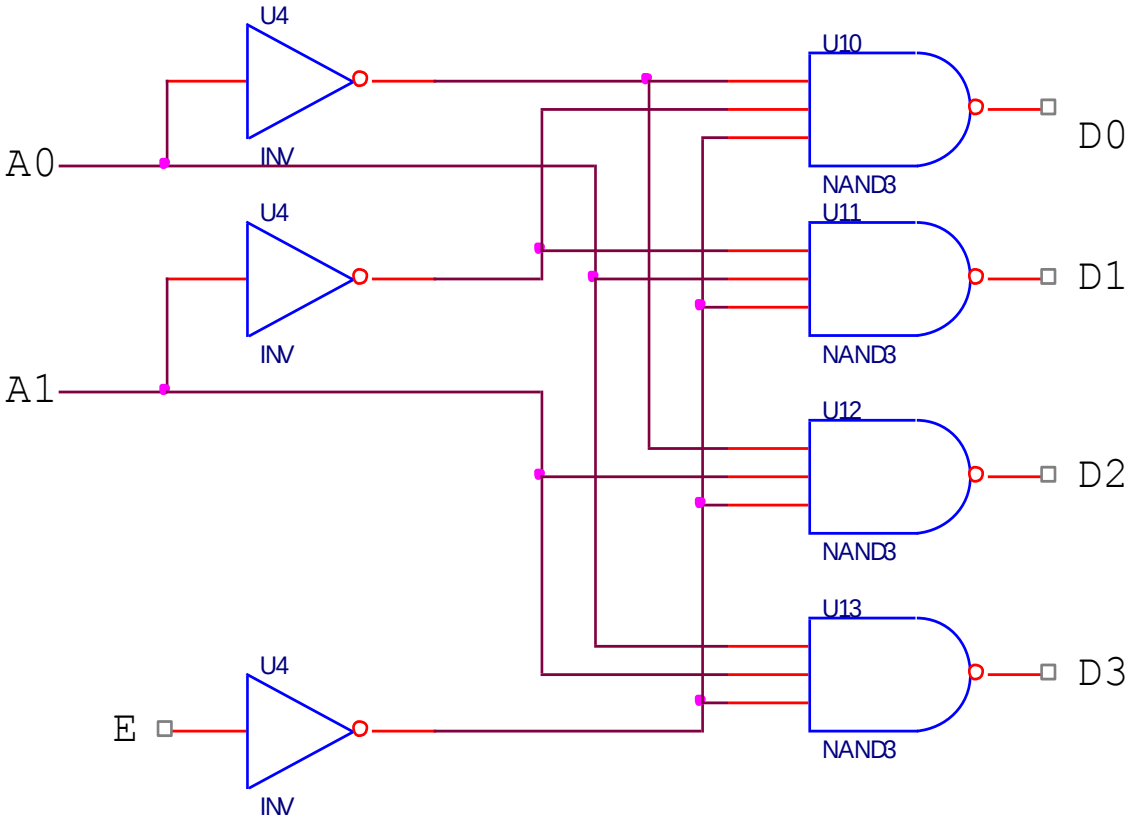
Mạch giải mã 3-8

A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
0	**0**	**0**	1	0	0	0	0	0	0	0
0	**0**	**1**	0	1	0	0	0	0	0	0
0	**1**	**0**	0	0	1	0	0	0	0	0
0	**1**	**1**	0	0	0	1	0	0	0	0
1	**0**	**0**	0	0	0	0	1	0	0	0
1	**0**	**1**	0	0	0	0	0	1	0	0
1	**1**	**0**	0	0	0	0	0	0	1	0
1	**1**	**1**	0	0	0	0	0	0	0	1

Sơ đồ mạch giải mã 3-8



Mạch giải mã dùng cổng NAND

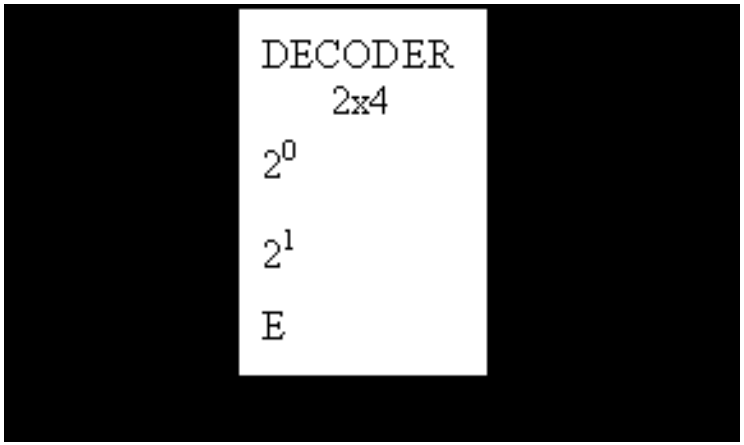


E	A1	A0	D0	D1	D2	D3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

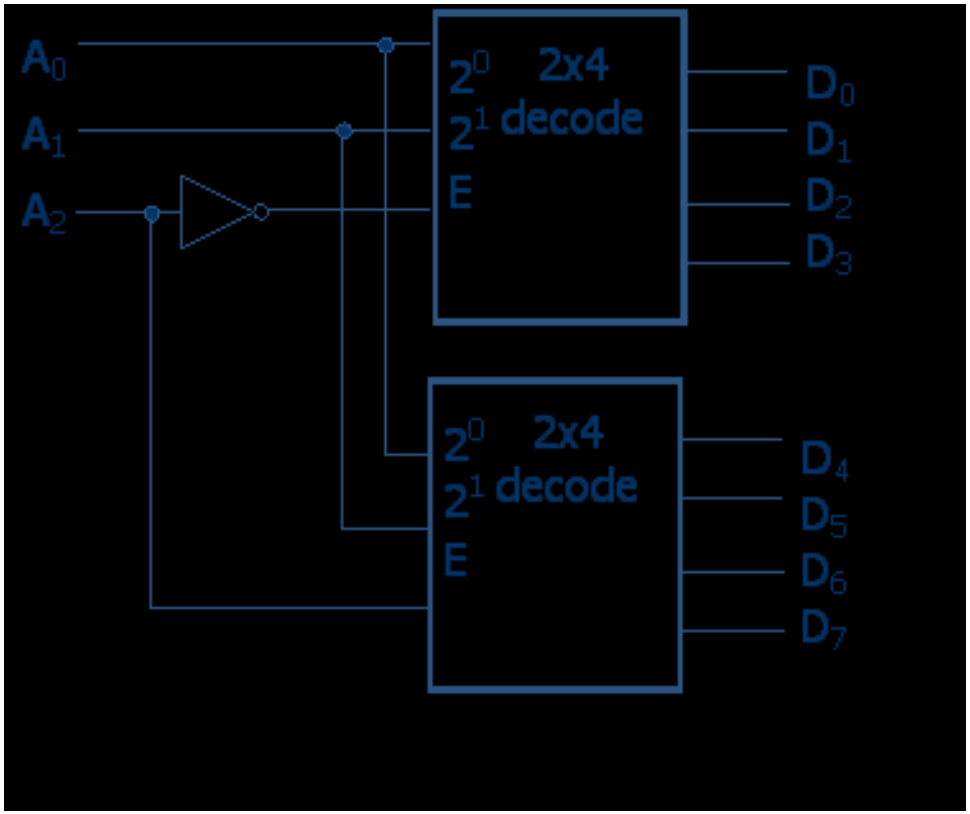
Mạch giải mã 2-4 với cổng NAND

Mở rộng mạch giải mã

Trong trường hợp cần mạch giải mã với kích cỡ lớn ta có thể ghép 2 hay nhiều mạch nhỏ hơn lại để được mạch cần thiết



Ký hiệu Decoder 2→4



Chương 5 – Mạch Tuần tự

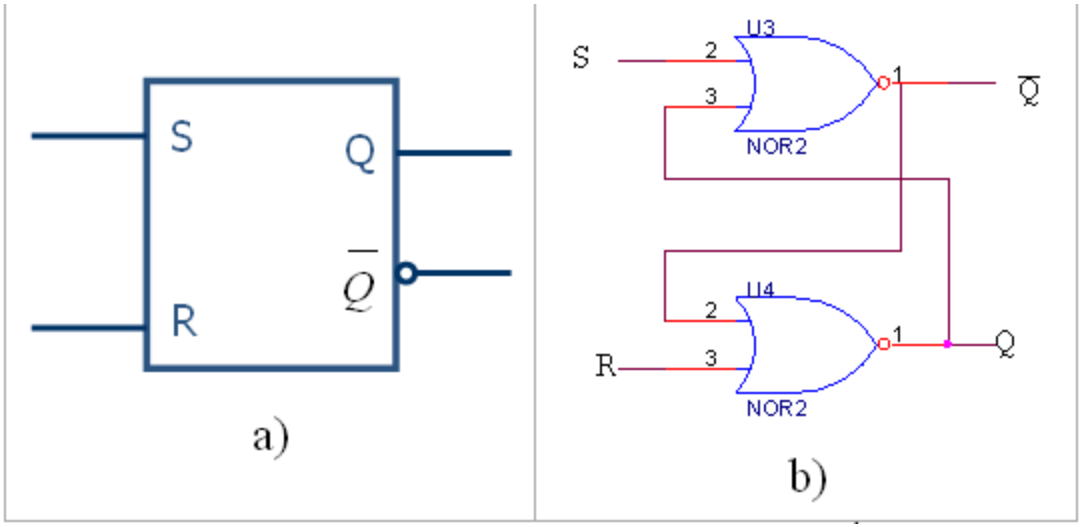
Xung đồng hồ

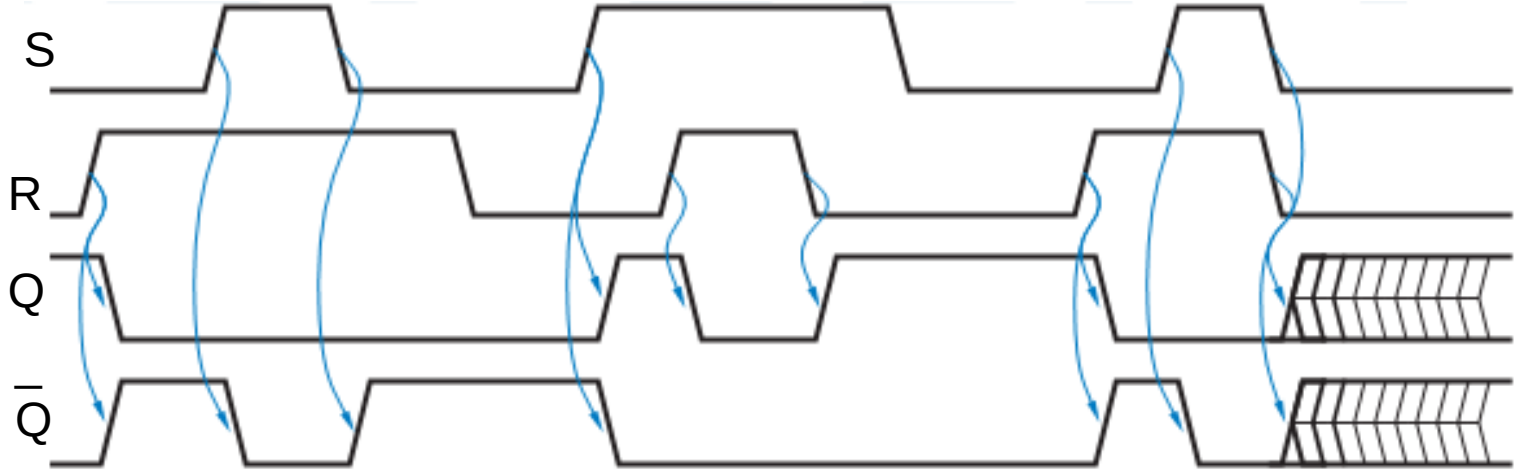
h.a) Đồng hồ (clock) –
bộ phát tần (impulse generator)
- thời gian chu kỳ đồng hồ (clock cycle time)
h.b – giản đồ thời gian của tín hiệu đồng hồ (4 tín hiệu thời gian cho các sự kiện khác nhau)
Sự sinh tín hiệu đồng hồ không cân xứng??

Mạch lật (Chốt - Latch)

Sơ đồ và ký hiệu chốt SR không dùng tín hiệu đồng hồ

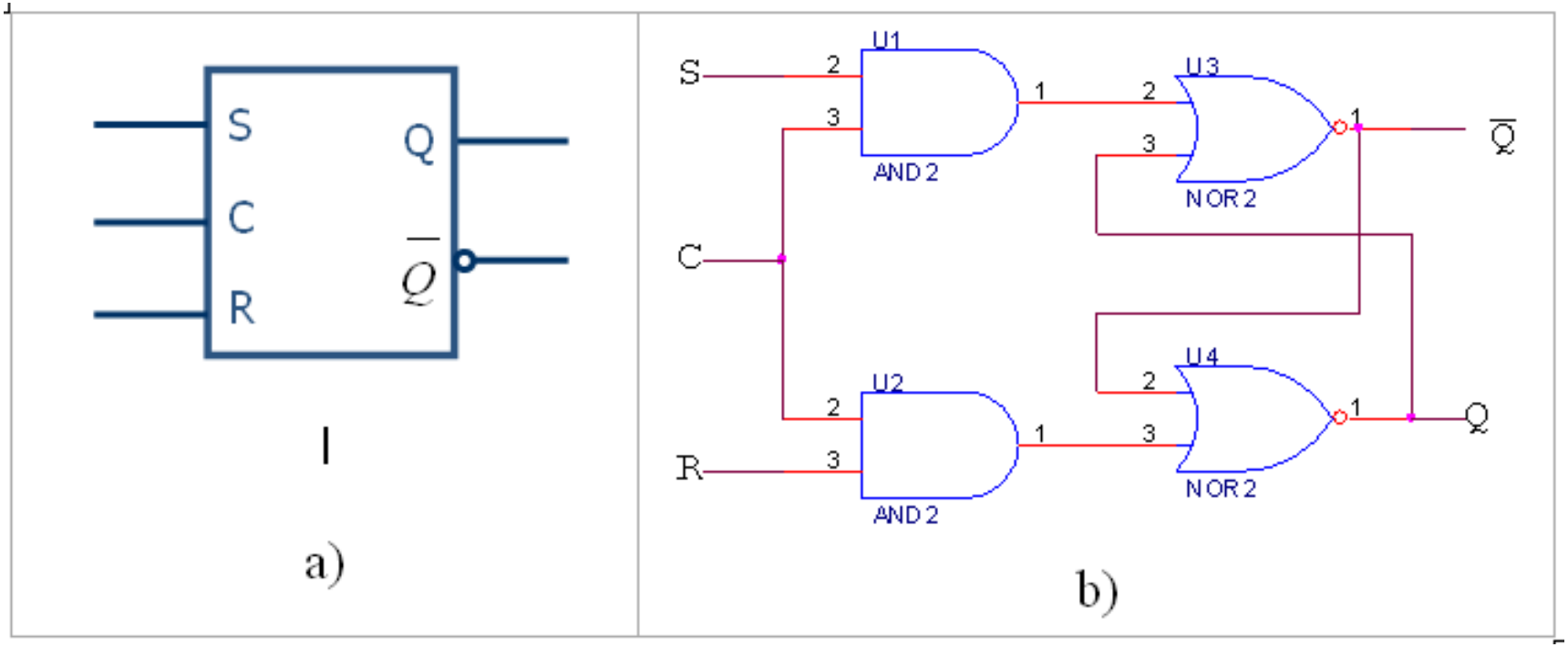
S	*R*	*Q(t+1)*	
0	0	Q(t)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	X	Indeterminate



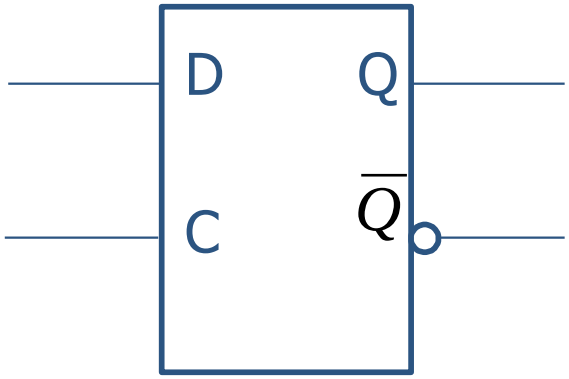


SR-latch

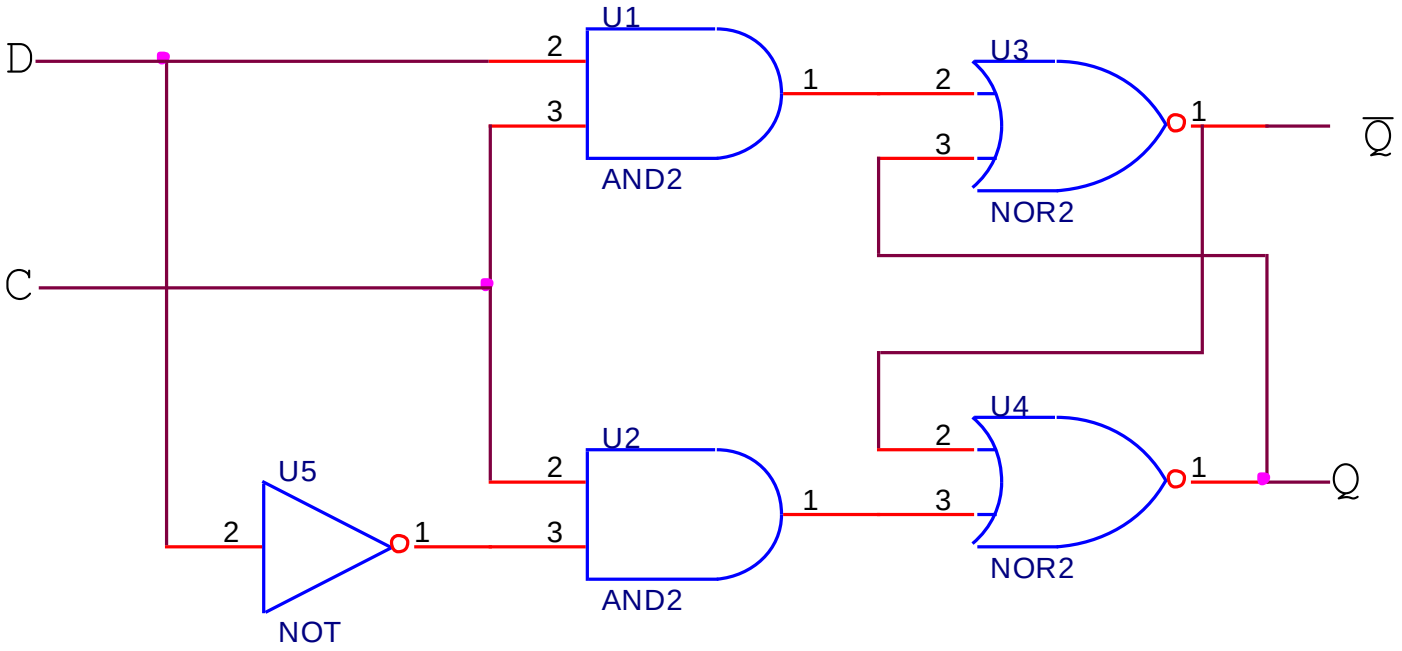
b) Mạch lật SR dùng tín hiệu đồng hồ



D latch

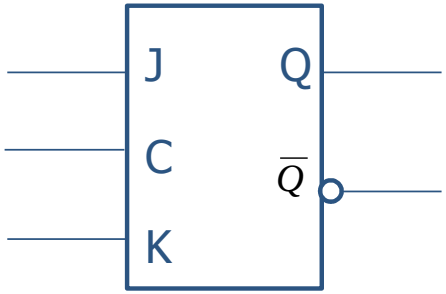


D	Q(t+1)	
0	0	Clear to 0
1	1	Set to 1



JK latch

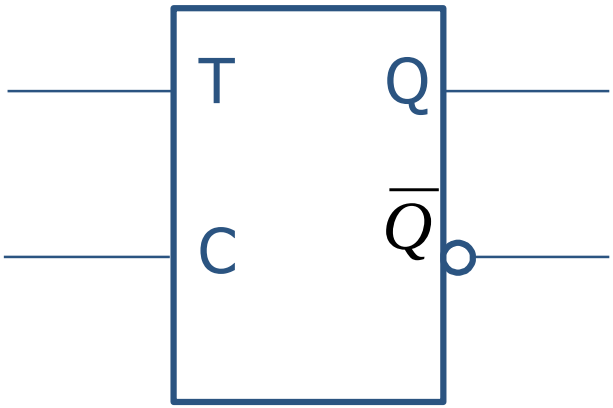
- Từ mạch lật SR
- Khắc phục nhược điểm của SR



J	*K*	*Q(t+1)*	
0	0	Q(t)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	$\overline{Q}(t)$	Complement

T latch

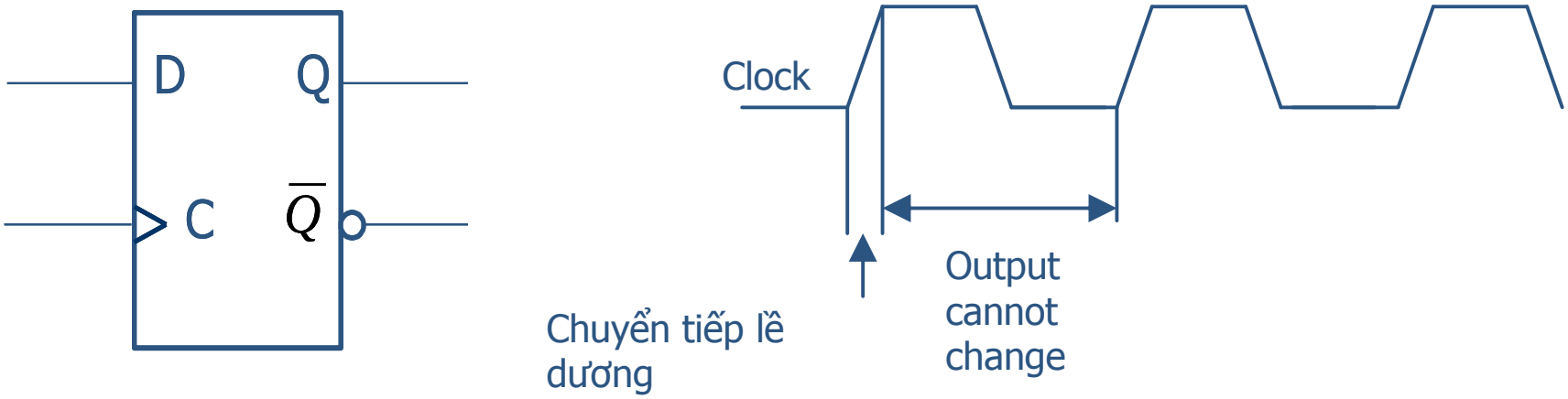
- Từ JK latch
- Nối J với K



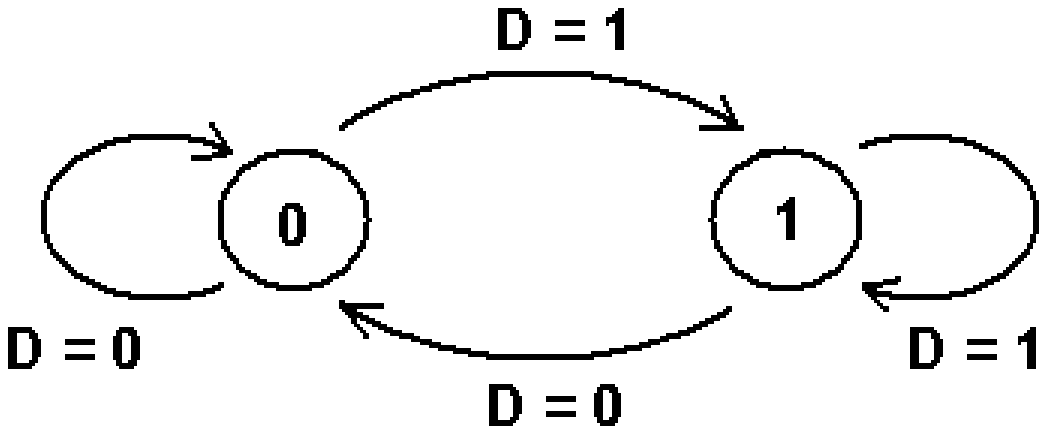
T	Q(t+1)	
0	Q(t)	No change
1	$\overline{Q}(t)$	Complement

Mạch lật lề (Flip-flop)

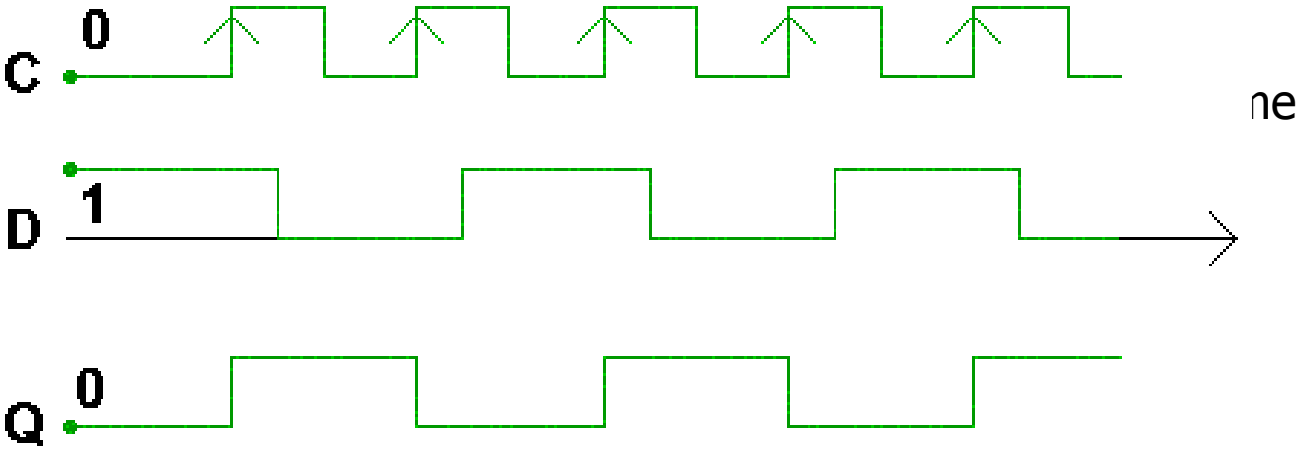
- ***Mạch lật kích thích bằng mức (level triggered),còn mạch lật lề kích thích bằng biên (edge triggered)***
- Flip-flop D với chuyển tiếp dương:



Flip-flop D



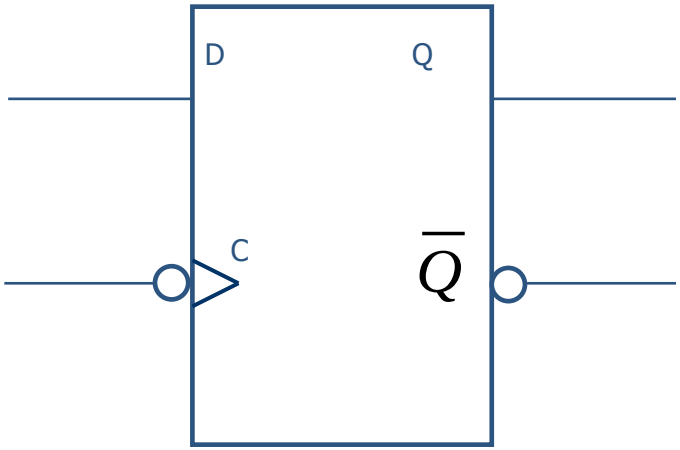
Biểu đồ trạng thái



Đồ thị dạng tín hiệu

Flip-flop D

- Flip-flop D với chuyển tiếp âm



Bảng kích thích của bốn mạch lật lề

SR

Q(t)	*Q(t+1)*	*S*	*R*
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

D

Q(t)	*Q(t+1)*	*D*
0	0	0
0	1	1
1	0	0
1	1	1

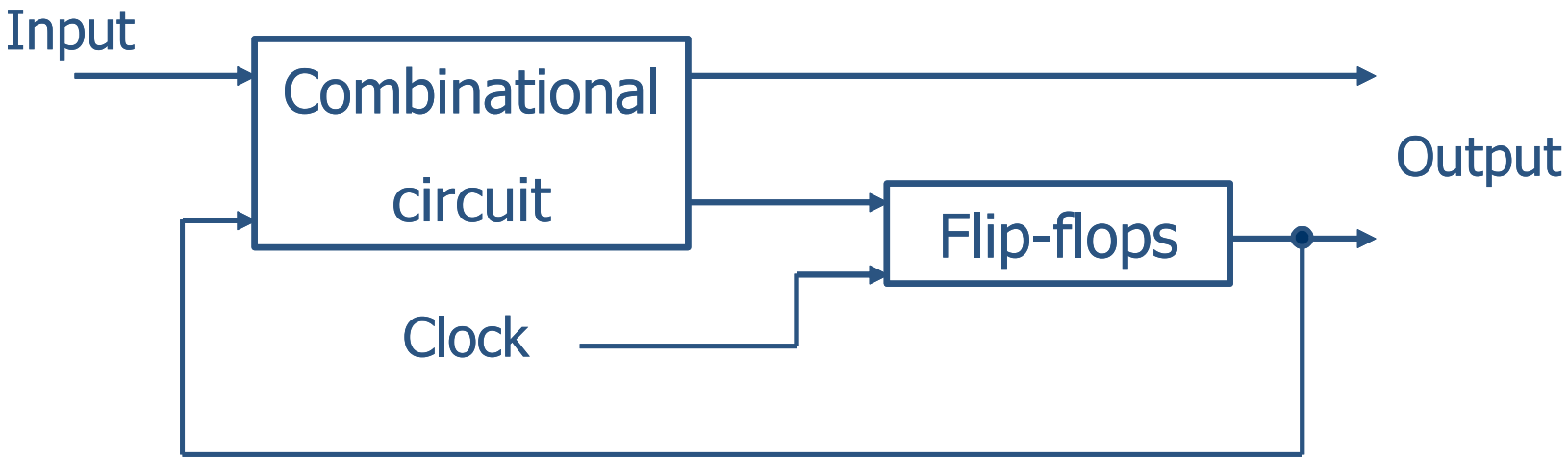
JK

Q(t)	*Q(t+1)*	*J*	*K*
0	0	0	X
0	1	1	x
1	0	x	1
1	1	X	0

T

Q(t)	*Q(t+1)*	*T*
0	0	0
0	1	1
1	0	1
1	1	0

Mạch tuần tự



❑ Qui trình thiết kế mạch tuần tự

- Bước 1: Chuyển đặc tả mạch sang lược đồ trạng thái
- Bước 2: lược đồ trạng thái => bảng trạng thái
- Bước 3: Từ bảng trạng thái viết hàm cho các ngõ nhập của Flip-flops
- Bước 4: vẽ sơ đồ mạch

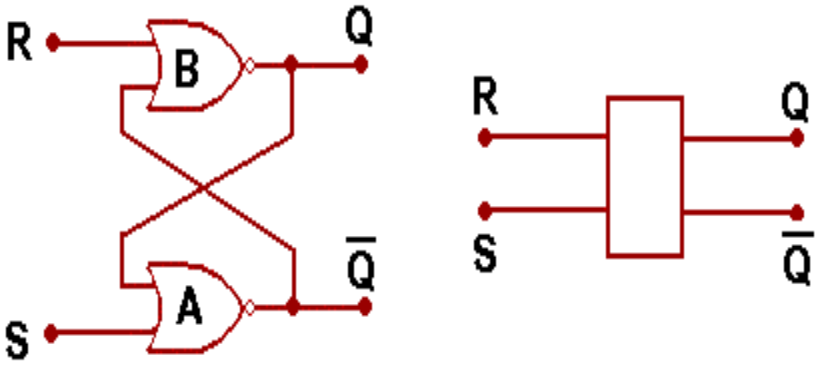
Ví dụ thiết kế mạch tuần tự

- Thiết kế mạch tuần tự dùng mạch lật SR. Khi ngõ nhập x=0, trạng thái mạch lật lề không thay đổi, ngõ xuất y=0. Khi x=1, dãy trạng thái là 11,10,01,00 và lặp lại còn ngõ xuất y sẽ có giá trị là 1 khi số bit trạng thái mạch lật lề bằng 1 là lẻ, các trường hợp còn lại thì bằng 0.

THANH GHI

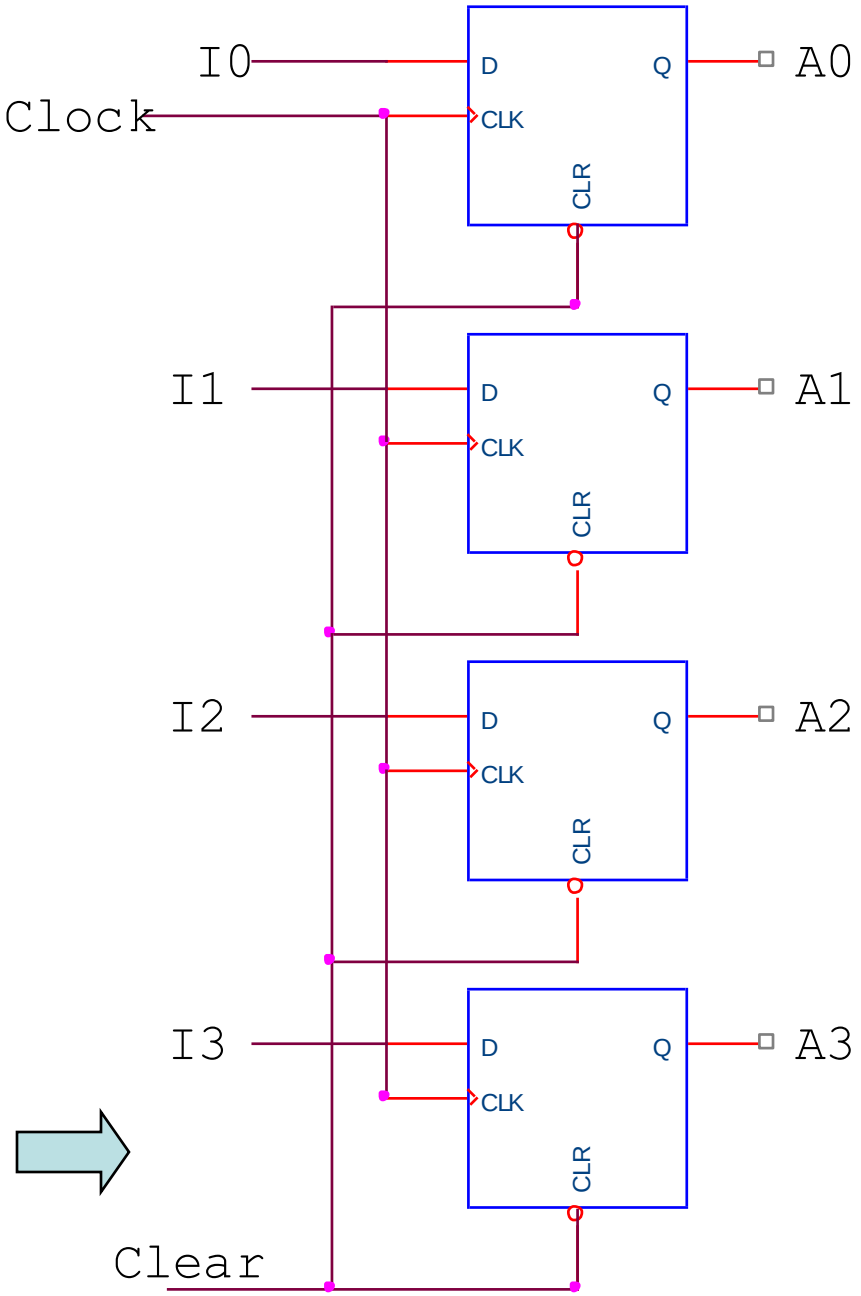
- Thanh ghi là một nhóm các mạch lật (mỗi mạch lưu 1 bit dữ liệu) và các cổng tác dộng đến chuyển tiếp của nó
- Thanh ghi đơn giản nhất -chốt RS

Sơ đồ, ký hiệu chốt RS



S	*R*	*Q(t+1)*
0	0	Q(t) No change
0	1	0 Clear to 0
1	0	1 Set to 1
1	1	X Indeterminate

- **Thanh ghi nạp song song**

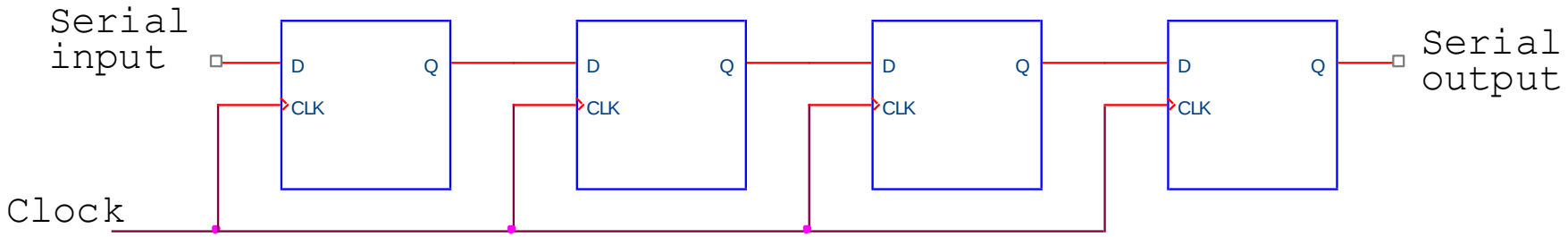


Thanh ghi nạp song song

- Thanh ghi 4 bit

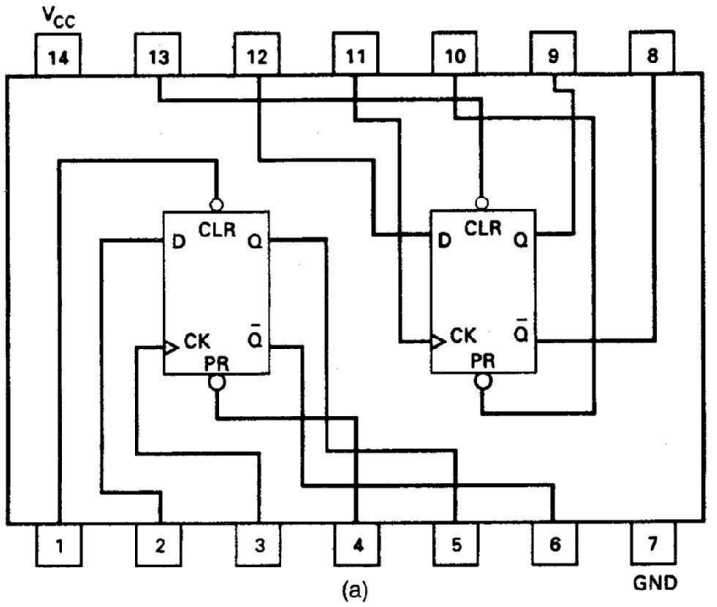
Thanh ghi dịch 4 bit

- Thanh ghi có khả năng dịch thông tin nhị phân theo một hoặc cả 2 hướng được gọi là thanh ghi dịch

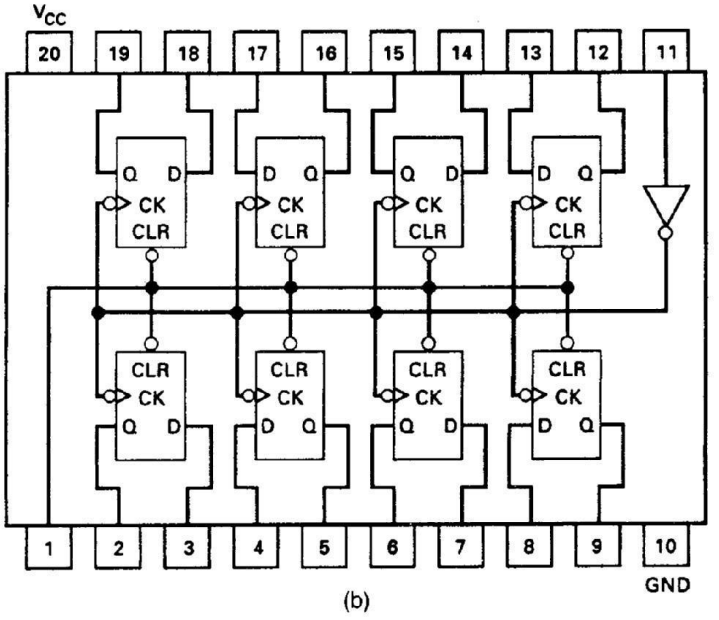


- Serial input – cho dữ liệu đi vào
- Serial ouput – cho dữ liệu ra
- Clock – xung đồng hồ để điều khiển các thao tác dịch

- IC Flip-Flop từ đó có thể tạo các thanh ghi



(a)



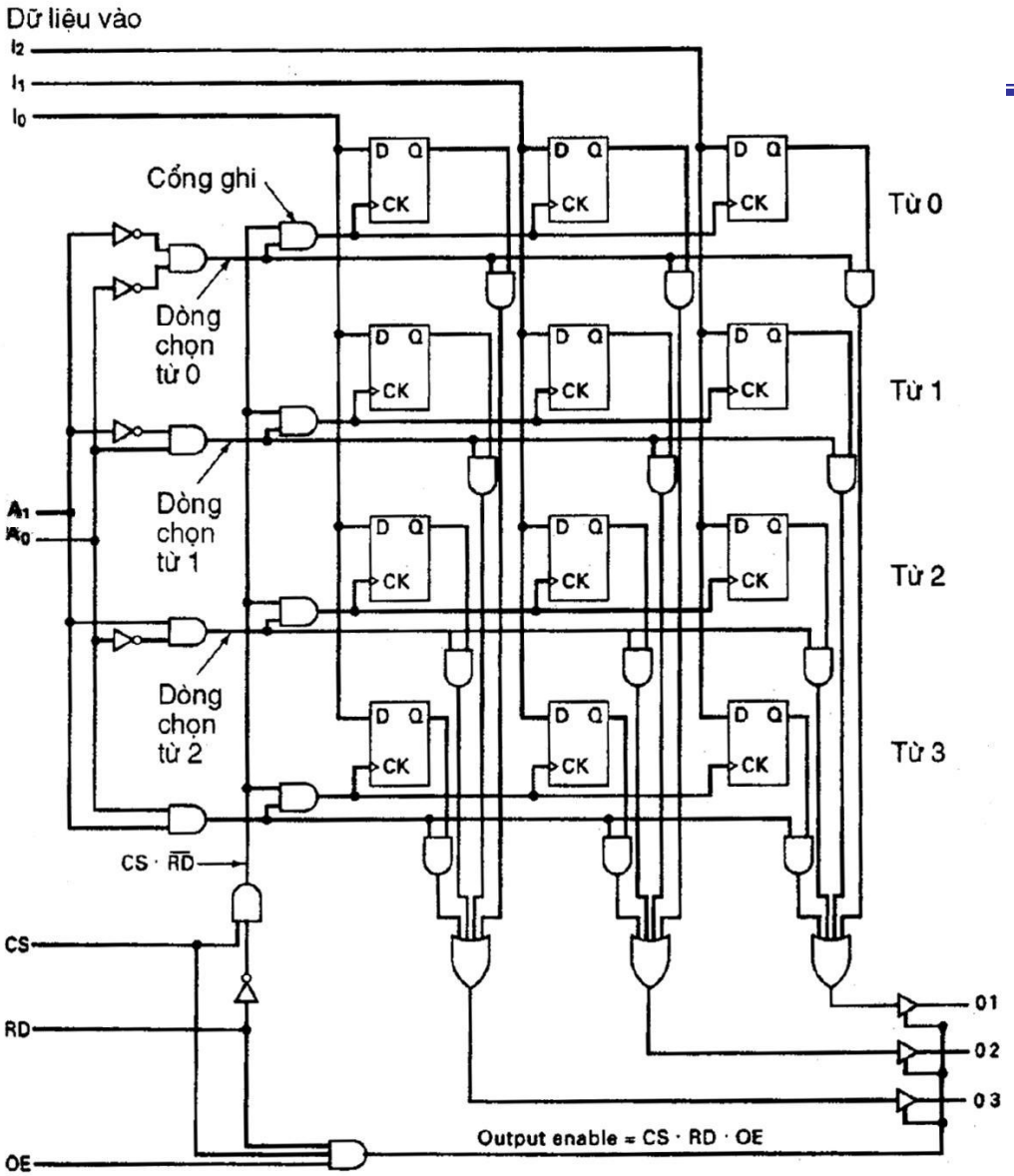
(b)

(a) Flip-flop D kép 7474. (b) Flip-flop bát phân 74273.

BỘ NHỚ

- Bộ nhớ ***(memory)*** là thành phần lưu trữ chương trình và dữ liệu trong máy tính.
- **Bit** – Đơn vị cơ bản của bộ nhớ là số nhị phân, gọi là ***bit***.
- **Địa chỉ bộ nhớ** - Bộ nhớ gồm một số ô (hoặc vị trí), mỗi ô ***(cell)*** có thể chứa một mẩu thông tin. Mỗi ô gắn một con số gọi là địa chỉ ***(address)***, qua đó chương trình có thể tham chiếu nó.
  - Tất cả các ô trong bộ nhớ đều chứa cùng số bit.
  - Các ô kế cận có địa chỉ liên tiếp nhau.
- Ô là đơn vị có thể lập địa chỉ nhỏ nhất -> chuẩn hóa ô 8 bit, gọi là ***byte***. Byte nhóm lại thành từ ***(word)*** – *hầu hết các lệnh được thực hiện trên từ.*

Tổ chức bộ nhớ



Sơ đồ logic cho bộ nhớ 4 x 3. Mỗi hàng là một trong bốn từ 3 bit. Hoạt động đọc/ghi luôn đọc/ghi nguyên cả từ.

Chương 6 – Kiến trúc bộ lệnh

6.1. Phân loại kiến trúc bộ lệnh

- kiến trúc ngăn xếp (stack),
- kiến trúc thanh ghi tích lũy (Accumulator)
- kiến trúc thanh ghi đa dụng GPRA(general-purpose register architecture).

Ví dụ phép tính C = A + B được dùng trong các kiểu kiến trúc:

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R1,B	Load R2,B
Add	Store C	Store C,R1	Add R3,R1,R2
Pop C			Store C,R3

Kiểu kiến trúc GPR

- Ưu điểm
  - Dùng thanh ghi, một dạng lưu trữ trong của CPU có tốc độ nhanh hơn bộ nhớ ngoài
  - Trình tự thực hiện lệnh có thể ở mọi thứ tự
  - Dùng thanh ghi để lưu các biến và như vậy sẽ giảm thâm nhập đến bộ nhớ => chương trình sẽ nhanh hơn
- Nhược điểm
  - Lệnh dài
  - Số lượng thanh ghi bị giới hạn
- Ngăn xếp (Stack) ?
- Thanh ghi tích luỹ (Accumulator Register) ?

Kiểu kiến trúc thanh ghi đa dụng

- lệnh có 2 toán hạng

  ADD A, B

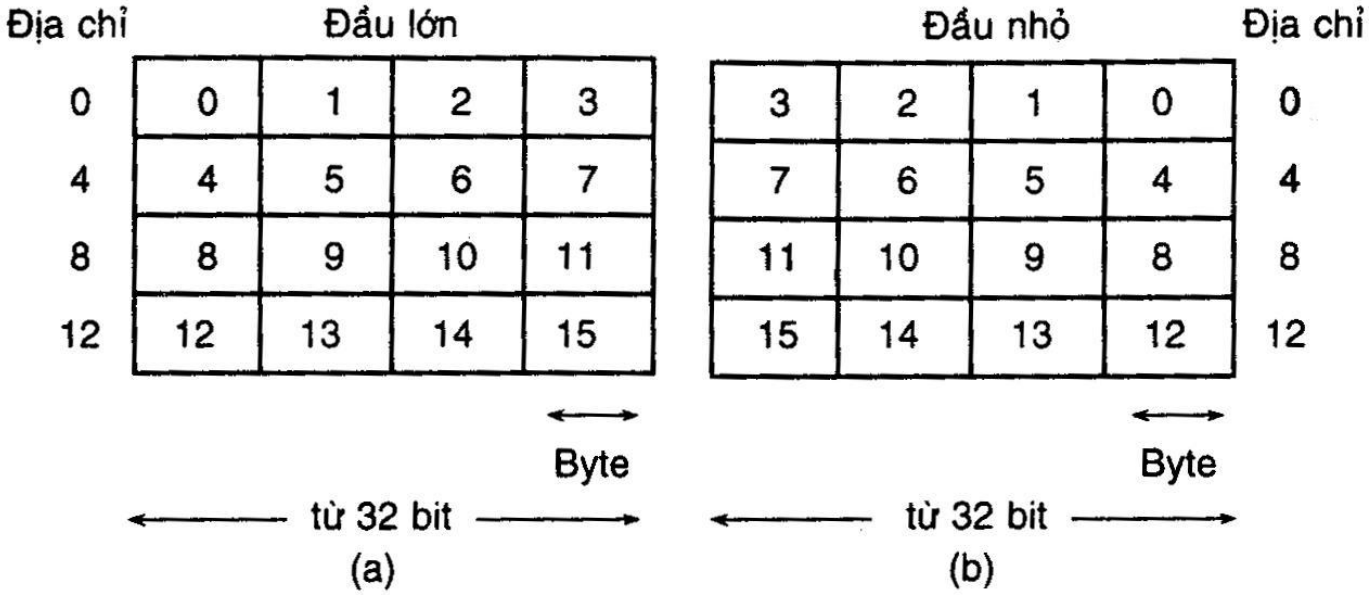
- lệnh có 3 toán hạng

  ADD A, B, C

- Số toán hạng bộ nhớ có thể thay đổi từ 0 tới 3
- Các loại toán hạng
  - thanh ghi-thanh ghi (kiểu này còn được gọi nạp - lưu trữ),
  - thanh ghi - bộ nhớ
  - bộ nhớ - bộ nhớ.

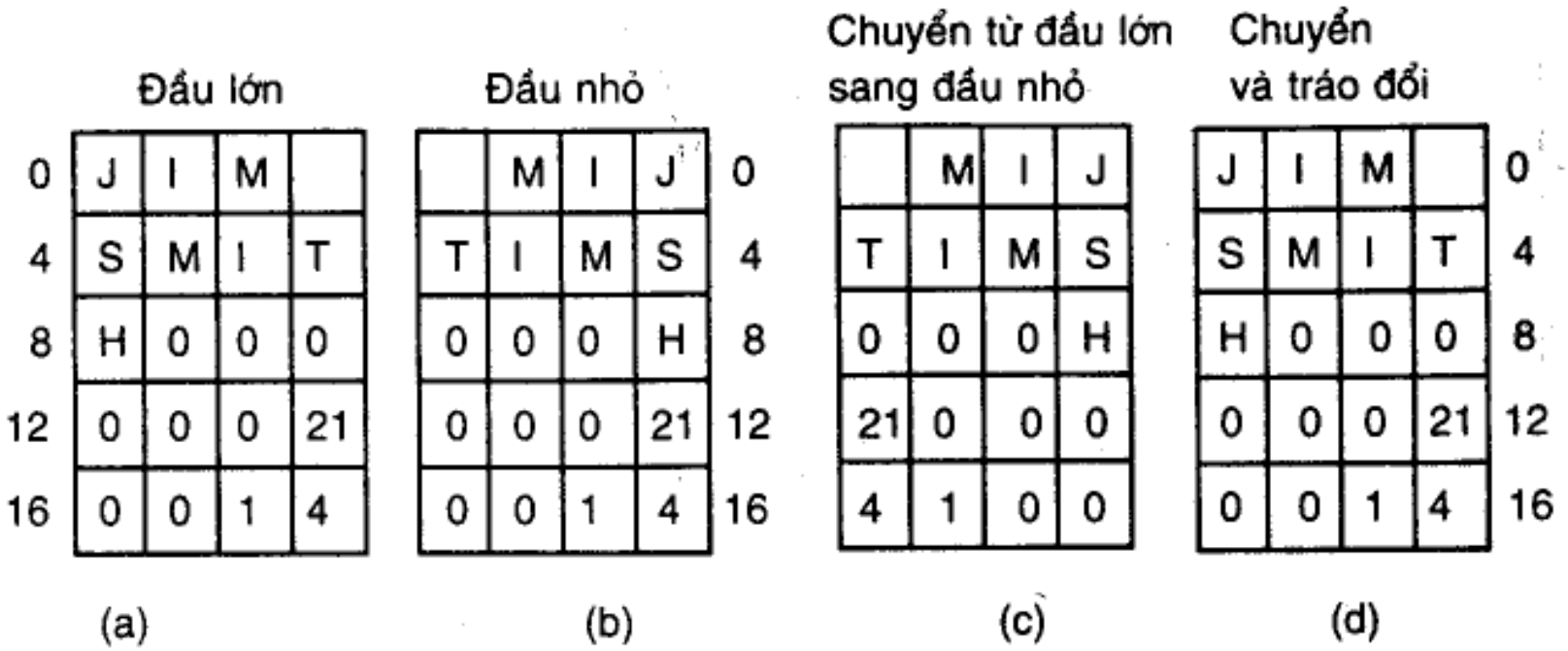
6.2. Địa chỉ bộ nhớ

- Các khái niệm:
  - Memory, bit, cell, address, byte, word
- **Sắp xếp thứ tự byte**
  - **Có vấn đề gì không trong cách sắp xếp thứ tự byte**



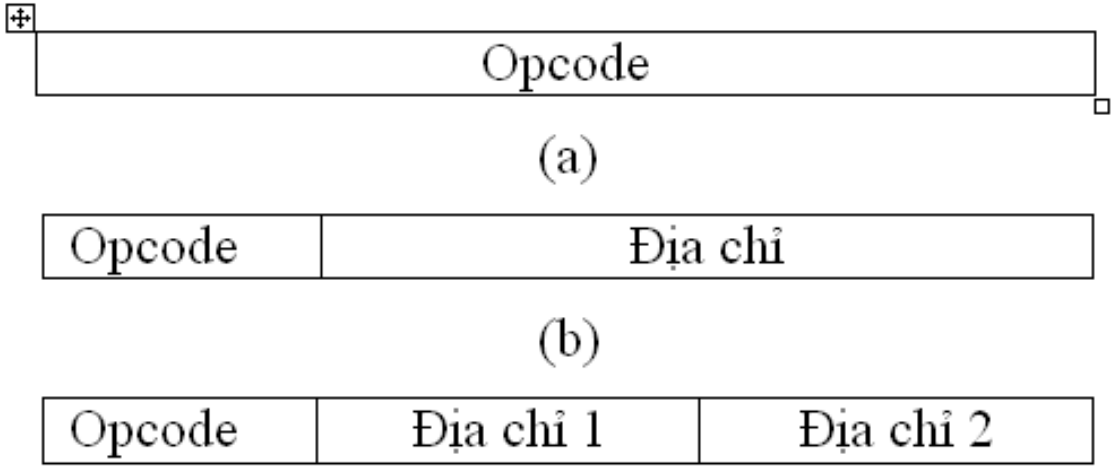
Vấn đề thứ tự byte

VD: Biểu diễn JIM SMITH, 21 tuổi, phòng 260



6.3. Mã hóa tập lệnh

- Các trường mã hóa:
  - ***mã tác vụ* (*operation code*):** Opcode
  - Địa chỉ



Các tiêu chuẩn thiết kế dạng thức lệnh

- ***Có 4 tiêu chuẩn thiết kế:***
  - Mã lệnh ngắn ưu việt hơn mã lệnh dài
  - Độ dài mã lệnh đủ để biểu diễn tất cả phép toán mong muốn
  - độ dài word của máy bằng bội số nguyên của độ dài ký tự
  - số BIT trong trường địa chỉ càng ngắn càng tốt

Ví dụ thiết kế máy với ký tự 8 bit và bộ nhớ chính chứa 2^{16} ký tự

Bộ nhớ dung lượng : $2^{16} * 8 = 2^{19}$ bit

+ Ô nhớ kích thước 8 bit =>

Số ô nhớ: $2^{19} / 8 = 2^{16}$ [redacted]ường địa chỉ cần 16 bit

+ Ô nhớ kích thước 32 bit

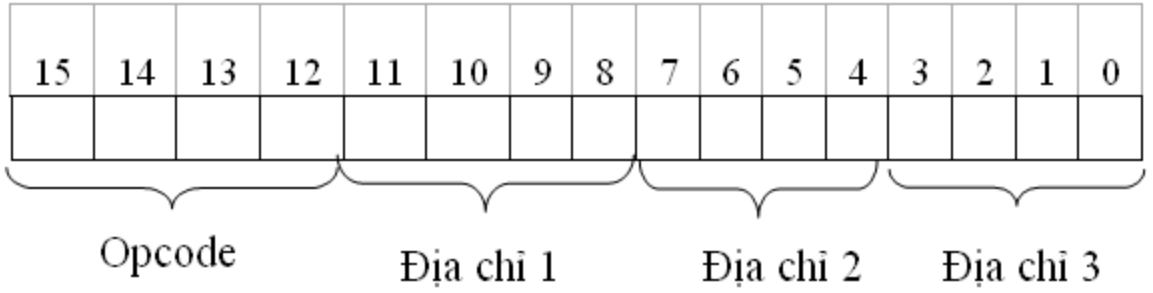
Số ô nhớ: $2^{19} / 32 = 2^{14}$ [redacted]ường địa chỉ cần 14 bit

	0 8
0	Word 0
1	
2^{16}-1	

	0			31
0	Word 0	Word 1	Word 2	Word 3
1				
2^{14}-1				

Opcode mở rộng

Ví dụ một máy tính có lệnh dài 16 bit :



- Lệnh (n+k) bit với opcode chiếm k bit và địa chỉ chiếm n bit.

Ví dụ: 15 lệnh ba địa chỉ

16 bit

opcode 4 bit

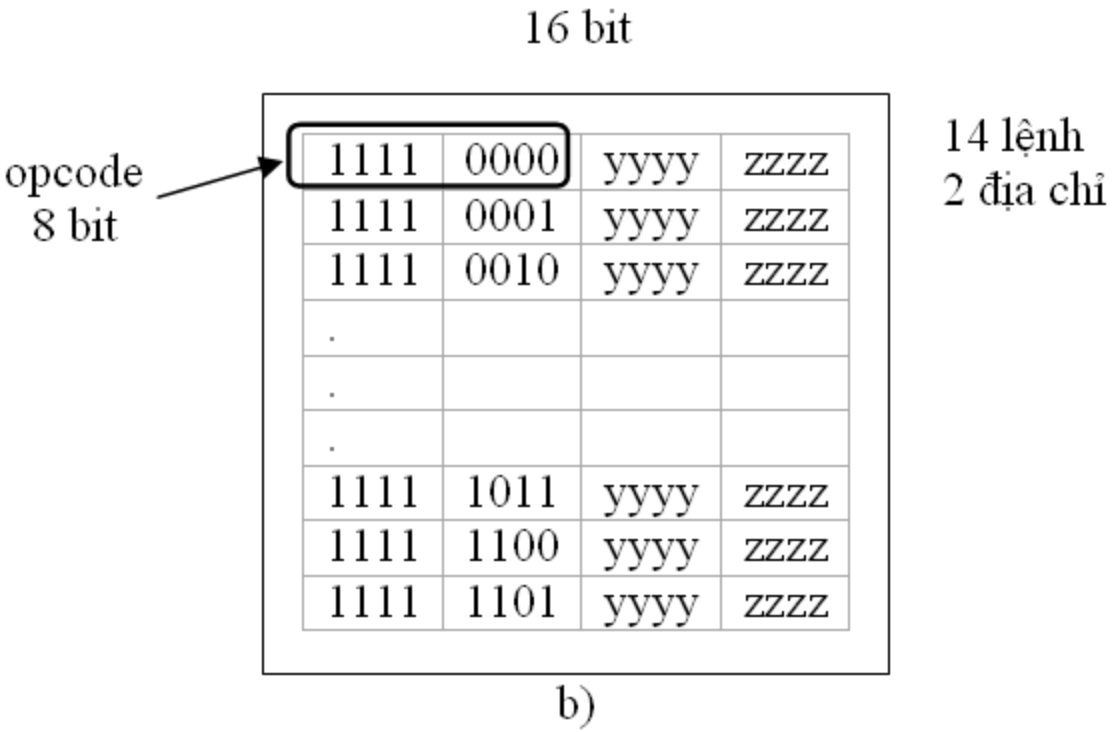
0000	xxxx	yyyy	zzzz
0001	xxxx	yyyy	zzzz
0010	xxxx	yyyy	zzzz
.			
.			
.			
1100	xxxx	yyyy	zzzz
1101	xxxx	yyyy	zzzz
1110	xxxx	yyyy	zzzz

15 lệnh 3 địa chỉ

a)

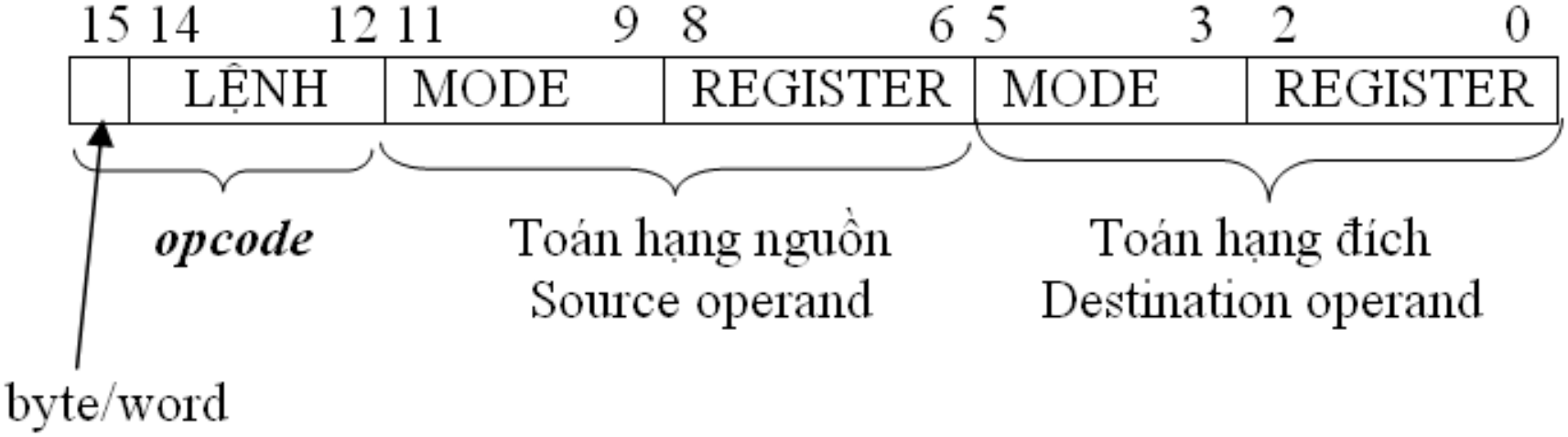
Opcode mở rộng

- 14 lệnh hai địa chỉ



dạng thức lệnh PDP-11

- Mã hóa lệnh trên máy PDP-11



- tám cách trên PDP-11
- opcode mở rộng có dạng x111
- các lệnh một toán hạng
  - opcode 10 bit: 4 bit opcode và 6 bit của trường toán hạng nguồn
  - mode/register 6 bit

Họ Intel 8088/80286/80386/Pentium

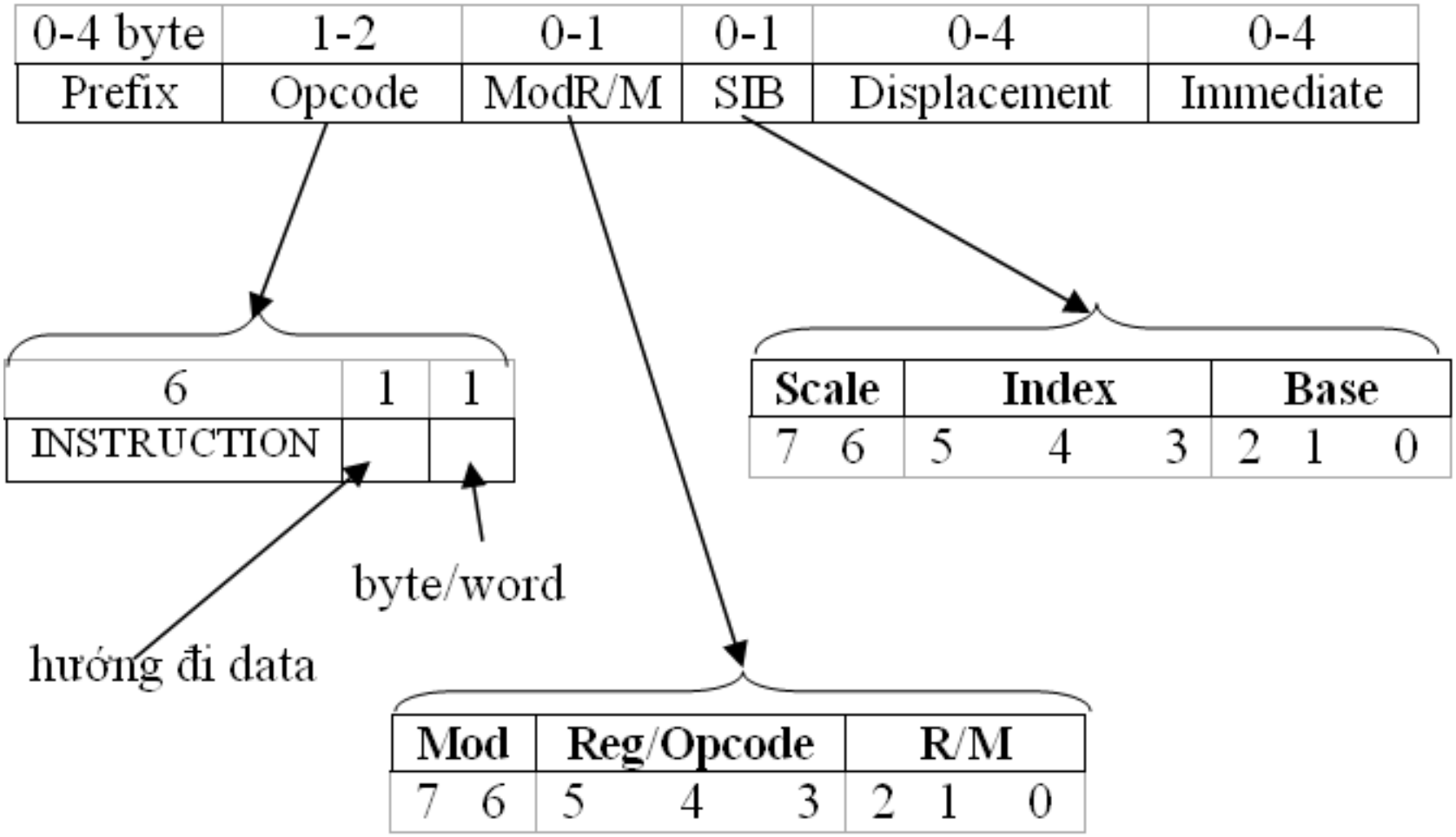
- Dạng thức lệnh của các máy tính Intel:
  - Cấu tạo phức tạp
  - kế thừa từ nhiều thế hệ
  - bốn cách lập địa chỉ toán hạng (so với tám cách trên PDP-11)

CPU	PREFIX	OPCODE	MODE	SIB	DISPLACEMENT	IMMEDIATE
8088	0-3	1	0-1	0	0-2	0-2
80286	0-3	1	0-1	0	0-2	0-2
80386	0-4	1-2	0-1	0-1	0-4	0-4
Pentium	0-4	1-2	0-1	0-1	0-4	0-4

PREFIX byte:

- LOCK prefix: để đảm bảo việc dành riêng vùng nhớ chia sẻ trong môi trường đa bộ xử lý
- REPeat prefix: đặc trưng cho một chuỗi phép toán được lập đi lập lại

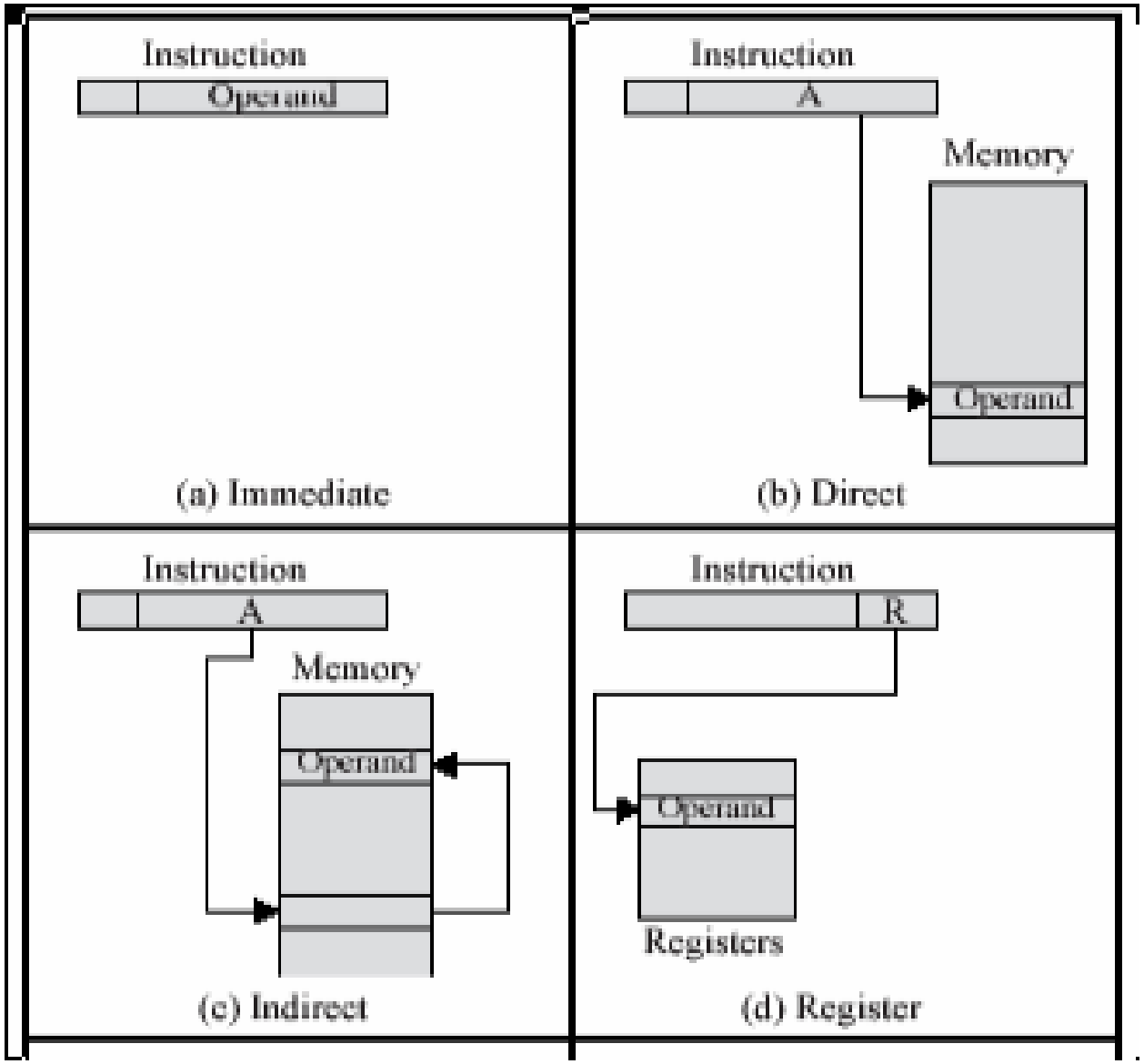
Format lệnh Pentium

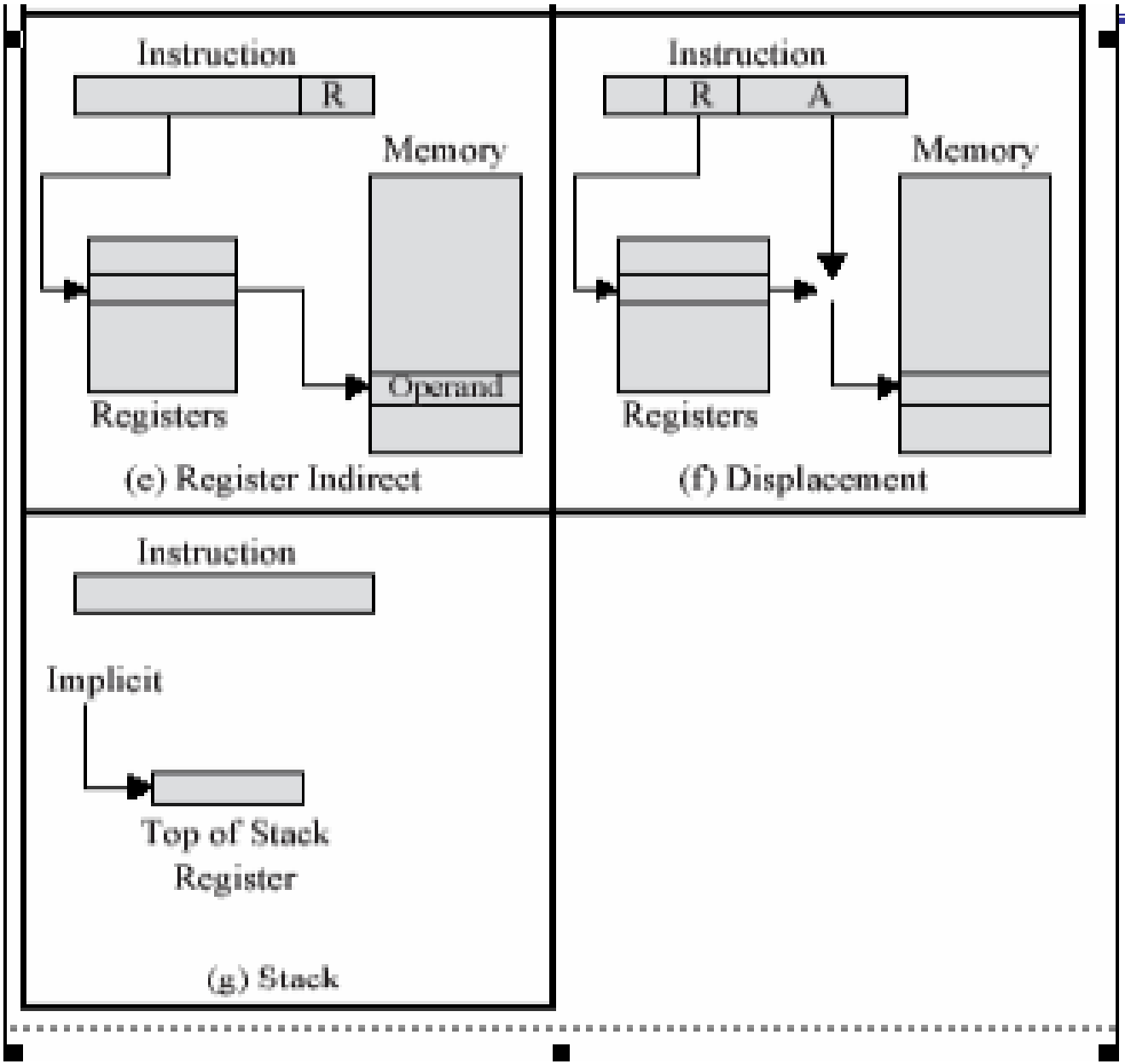


Các chế độ lập địa chỉ

- Địa chỉ tức thời – Immediate
- Địa chỉ trực tiếp – Direct
- Địa chỉ gián tiếp – Indirect
- Địa chỉ thanh ghi – Register
- Địa chỉ gián tiếp thanh ghi – Register inderect
- Địa chỉ dịch chuyển – Displacement
- Địa chỉ ngăn xếp - Stack

Các chế độ lập địa chỉ




Instruction
R
Memory
Operand
Registers
(e) Register Indirect
Instruction
R
A
Memory
Registers
(f) Displacement
Instruction
Implicit
Top of Stack
Register
(g) Stack

Cách tính địa chỉ thực

Chế độ	Cách tính	Ưu điểm	Khuyến điểm
Immediate	Operand = A	Không có tham chiếu bộ nhớ	Độ lớn toán hạng giới hạn
Direct	EA = A	Đơn giản	không gian địa chỉ giới hạn
Indirect	EA = (A)	không gian địa chỉ lớn	Tham chiếu bộ nhớ phức tạp
Register	EA = R	Không có tham chiếu bộ nhớ	không gian địa chỉ giới hạn
Register indirect	EA = (R)	không gian địa chỉ lớn	Tham chiếu bộ nhớ phụ
Displacement	EA = A + (R)	Linh động	Phức tạp
Stack	EA= đầu của ngăn xếp	Không có tham chiếu bộ nhớ	Ứng dụng giới hạn

Các chế độ lập địa chỉ

- ***Lập địa chỉ tức thời (Immediate Addressing)***:
  - OPERAND = A
  - MOV R1, #4
- ***Lập địa chỉ trực tiếp (Direct Addressing)***:
  - EA = A
- ***Lập địa chỉ gián tiếp (Indirect Addressing)***
  - EA = (A)
  - một con trỏ (trong C++)
- ***Lập địa chỉ thanh ghi (Register Addressing)***
  - trỏ tới một thanh ghi
  - Các máy ngày nay được thiết kế có các thanh ghi vì lý do?

Các chế độ lập địa chỉ

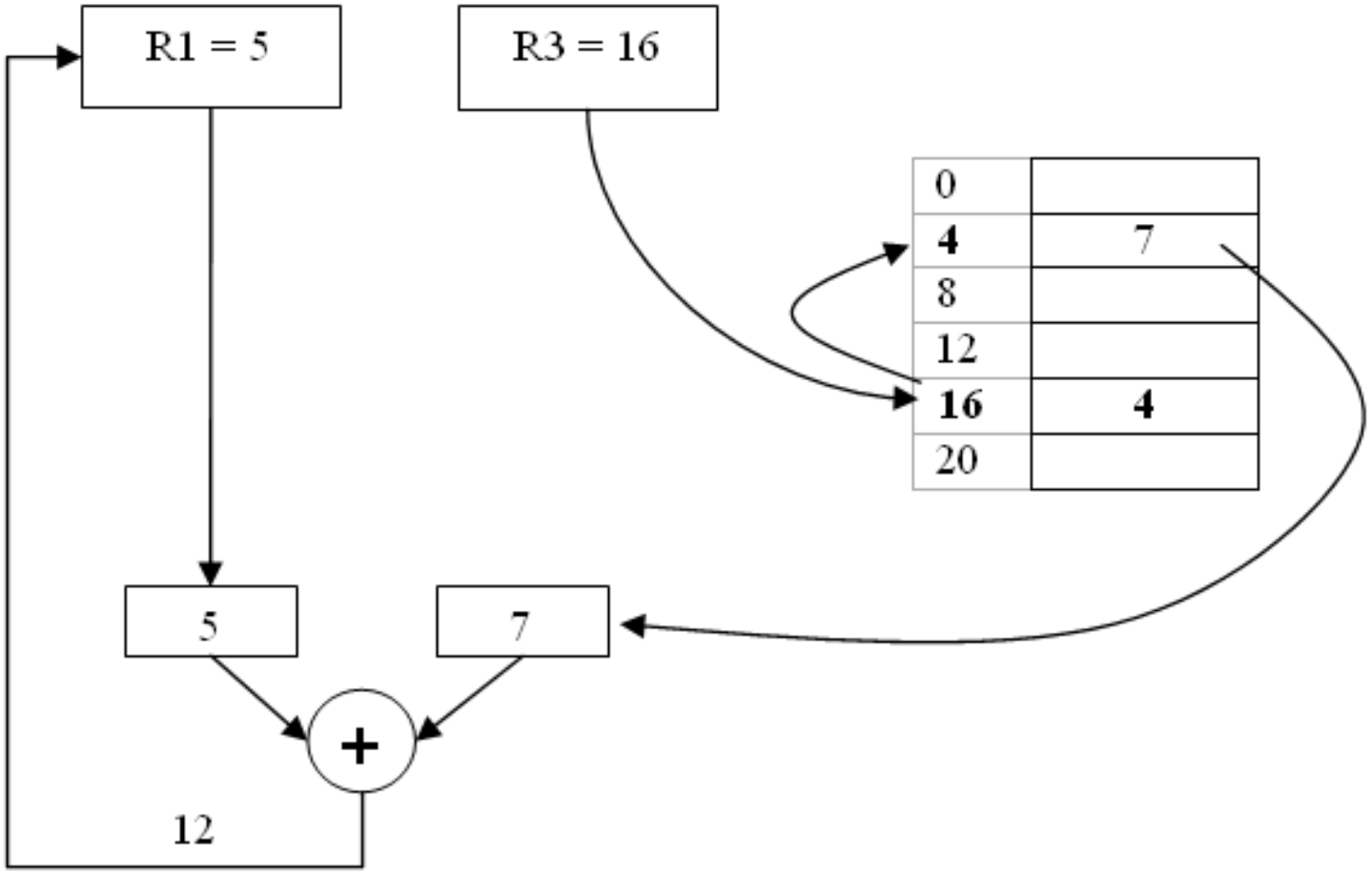
- ***Địa chỉ gián tiếp thanh ghi (Register Indirect)***
  - EA = (R)
- ***Địa chỉ Địa chỉ dịch chuyển – Displacement***
  - EA = A + (R)
- ***Địa chỉ ngăn xếp – Stack***
  - ***FILO (first in last out)***

VD:

Addressing mode	Example instruction	Meaning	When used
Register	`Add R4,R3`	`Regs[R4]←Regs[R4]+Regs[R3]`	When a value is in a register.
Immediate	`Add R4,#3`	`Regs[R4]←Regs[R4]+3`	For constants.
Displacement	`Add R4,100(R1)`	`Regs[R4]←Regs[R4]+Mem[100+Regs[R1]]`	Accessing local variables.
Register deferred or indirect	`Add R4,(R1)`	`Regs[R4]←Regs[R4]+Mem[Regs[R1]]`	Accessing using a pointer or a computed address.
Indexed	`Add R3,(R1 + R2)`	`Regs[R3]←Regs[R3]+Mem[Regs[R1]+Regs[R2]]`	Sometimes useful in array addressing: R1 = base of array; R2 = index amount.
Direct or absolute	`Add R1,(1001)`	`Regs[R1]←Regs[R1]+Mem[1001]`	Sometimes useful for accessing static data; address constant may need to be large.
Memory indirect or memory deferred	`Add R1,@(R3)`	`Regs[R1]←Regs[R1]+Mem[Mem[Regs[R3]]]`	If R3 is the address of a pointer *p*, then mode yields **p*.
Autoincrement	`Add R1,(R2)+`	`Regs[R1]←Regs[R1]+Mem[Regs[R2]]` `Regs[R2]←Regs[R2]+d`	Useful for stepping through arrays within a loop. R2 points to start of array; each reference increments R2 by size of an element, *d*.
Autodecrement	`Add R1,-(R2)`	`Regs[R2]←Regs[R2]-d` `Regs[R1]←Regs[R1]+Mem[Regs[R2]]`	Same use as autoincrement. Autodecrement/increment can also act as push/pop to implement a stack.
Scaled	`Add R1,100(R2)[R3]`	`Regs[R1]← Regs[R1]+Mem[100+Regs[R2]+Regs[R3]*d]`	Used to index arrays. May be applied to any indexed addressing mode in some machines.

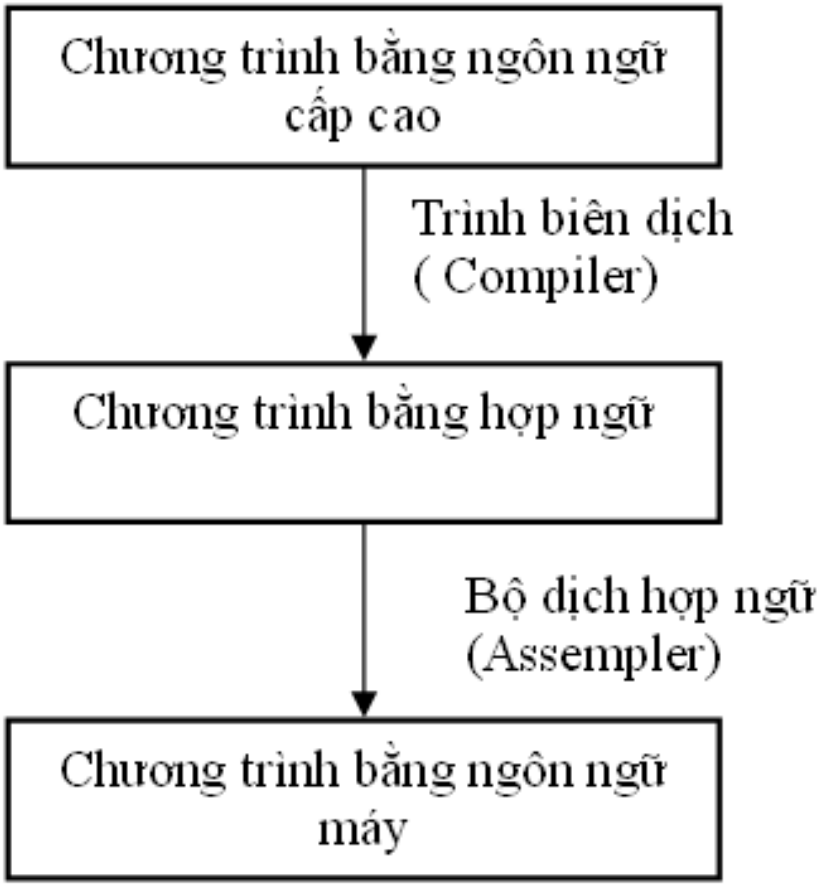
Ví dụ lệnh Add với tham chiếu bộ nhớ

- **Add R1, @(R3)**



6.4. Bộ lệnh

- Quá trình biên dịch ra ngôn ngữ máy



Nhóm lệnh truyền dữ liệu

- **MOVE Ri, Rj**
- **Một số ví dụ lệnh MOVE:**

Đích	Nguồn	Ví dụ	Giải thích
Bộ nhớ	Thanh ghi	MOVE 100H, AX	Chuyển nội dung trong AX vao vị trí nhớ 100H
Thanh ghi	Bộ nhớ	MOVE AX,MEM1	Chuyển nội dung trong vị trí nhớ MEM1 chỉ ra vào thanh ghi AX
Thanh ghi	Thanh ghi	MOVE AX, BX	Chuyển nội dung trong thanh ghi BX vào thanh ghi AX
Thanh ghi	Hằng số	MOVE AX, 0FFFFH	Chuyển giá trị hằng số ở hệ 16: FFFF vào thanh ghi AX, số 0 ở đầu để chỉ rõ FFFFH là một giá trị hằng chứ không phải là một nhãn

Nhóm lệnh truyền dữ liệu

- LOAD ***đích, nguồn***
  - ví dụ: LOAD Ri, M (địa chỉ) // Ri←M[địa chỉ]
- STORE ***đích, nguồn***
  - ví dụ: STORE M(địa chỉ), Ri // M[địa chỉ] ←Ri

Data movement operation	Meaning
MOVE	Move data (a word or a block) from a given source (a register or a memory) to a given destination
LOAD	Load data from memory to a register
STORE	Store data into memory from a register
PUSH	Store data from a register to stack
POP	Retrieve data from stack into a register

Nhóm lệnh tính toán số học

- ADD đích, nguồn // đích ← đích + nguồn
- SUB đích, nguồn // đích ← đích – nguồn
- Ví dụ:

```
ADD     AX, BX      // AX ←AX + BX
ADD     AL,74H      // AL ← AL + M[74H]
SUB     CL, AL      // CL ← CL – AL
SUB     AX, 0405H   // AX ← AX – 0405H
```

Nhóm lệnh tính toán số học

- Các lệnh tính toán số học cơ bản

Tên lệnh	Ý nghĩa
ADD	Cộng
ADDD	Cộng số có dấu chấm động, chính xác kép
SUB	Trừ
SUBD	Trừ số có dấu chấm động, chính xác kép
MUL	Nhân
DIV	Chia
INC	Tăng lên 1
DEC	Giảm đi 1
NEG	Đảo dấu toán hạng

Nhóm lệnh logic

- AND đích, nguồn
- OR đích, nguồn
- Ví dụ:

```
AND AL, BL
```

AL = 00001101B

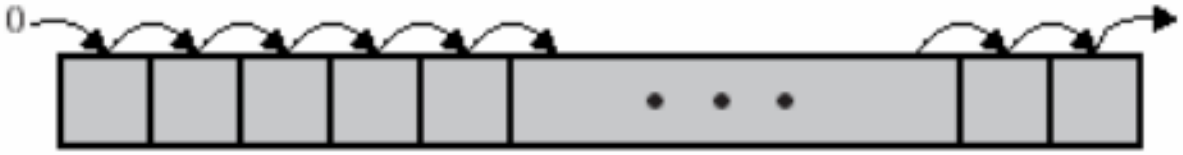
BL = 00110011B

=> AL = ***00000001B***

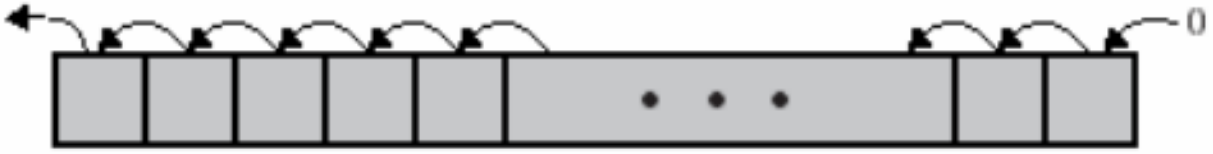
Nhóm các lệnh dịch chuyển số học hoặc logic (SHIFT)

- SRL (Shift Right Logical - dịch phải logic)
- SLL (Shift Left Logical - dịch trái logic)
- SRA (Shift Right Arithmetic - dịch phải số học)
- SLA (Shift Left Arithmetic – dịch trái số học)

Các lệnh dịch chuyển



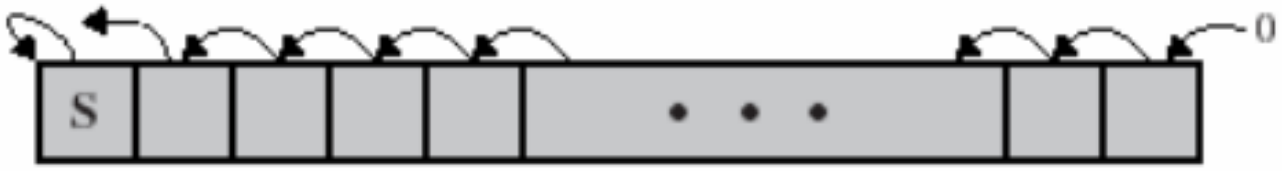
(a) Logical right shift



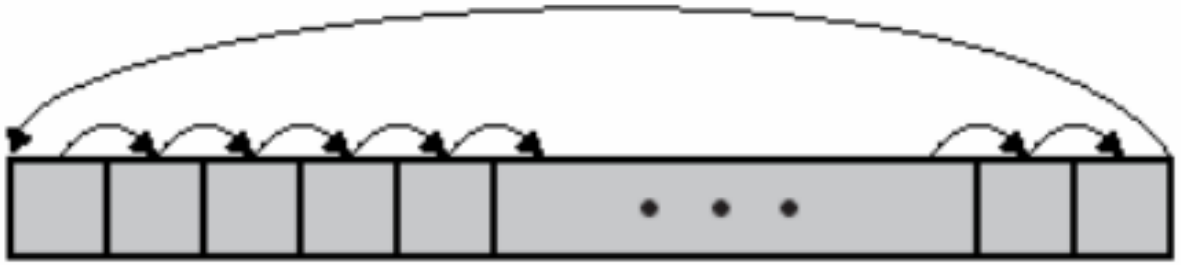
(b) Logical left shift

(c) Arithmetic right shift

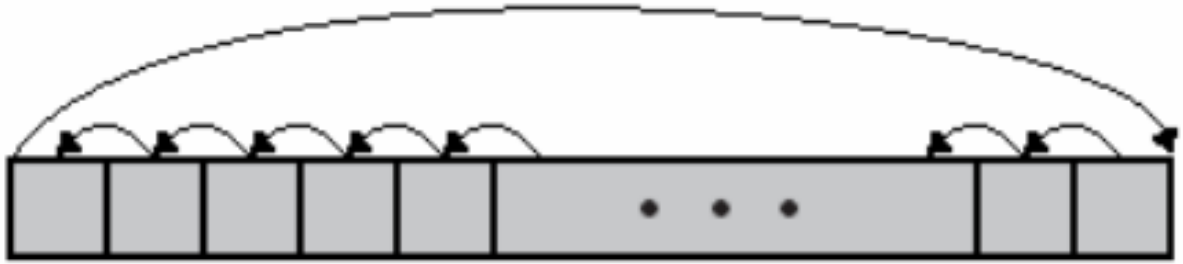
Các lệnh dịch chuyển



(d) Arithmetic left shift



(e) Right rotate

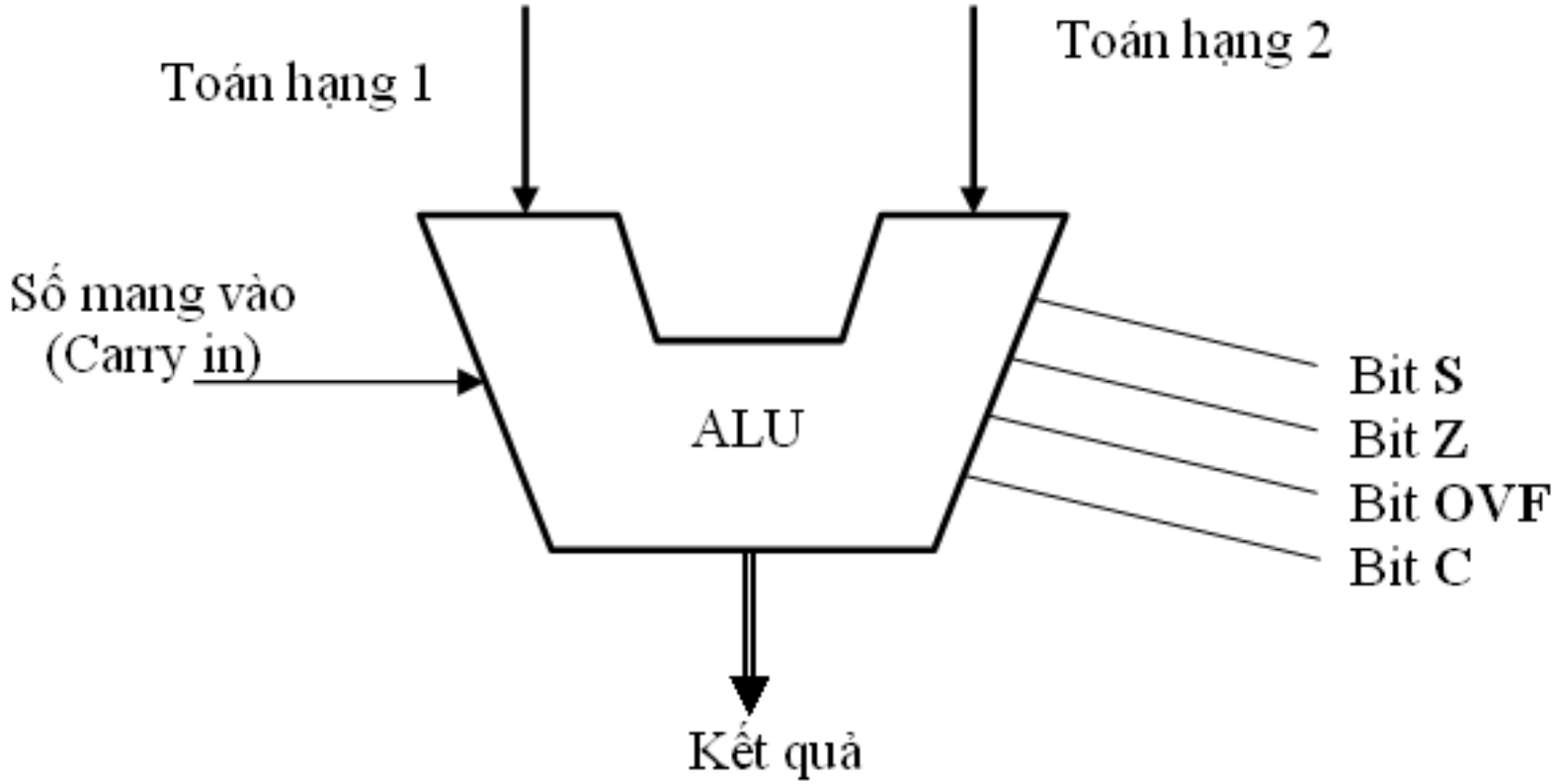


(f) Left rotate

Các lệnh có điều kiện và lệnh nhảy

Nếu <điều kiện> thì <chuỗi lệnh 1> nếu không <chuỗi lệnh 2>

(IF <condition> THEN <instructions1> ELSE <instructions2>)



Các lệnh có điều kiện và lệnh nhảy

Ví dụ:

```
        LOAD        R1, #100
Loop:   ADD         R0, (R2)+
        DECREMENT   R1
        BEQZ        R1, Loop // Giải thích
```

Thống kê sử dụng CPU

Loại lệnh	% sử dụng thời gian
Chuyển dữ liệu	43%
Điều khiển dòng chảy	23%
Tính toán số học	15%
So sánh	13%
Phép toán Logic	5%
Các lệnh khác	1%

Cấu trúc lệnh CISC và RISC

RISC	CISC
– Độ dài lệnh cố định (32 bit) – Sử dụng kiến trúc load-store các lệnh xử lý dữ liệu hoạt động chỉ trong thanh ghi và cách ly với các lệnh truy cập bộ nhớ – Một số lớn các thanh ghi đa dụng 32 bit – Có một số ít lệnh (thường dưới 100 lệnh) – Có một số ít các kiểu định vị – Có một số ít dạng lệnh (một hoặc hai) – Chỉ có các lệnh ghi hoặc đọc ô nhớ mới thâm nhập vào bộ nhớ.	– Kích thước tập lệnh thay đổi – Giá trị trong bộ nhớ được dùng như như toán hạng trong các chỉ lệnh xử lý dữ liệu – Có rất nhiều thanh ghi, nhưng hầu hết chỉ để sử dụng cho một mục đích riêng biệt nào đấy – Có rất nhiều lệnh (khoảng 500) – Có nhiều kiểu định vị (xem phần 6.3.4) – Có nhiều dạng lệnh – Có nhiều lệnh khác cũng thâm nhập vào bộ nhớ được
– Giải mã lệnh logic bằng kết nối phần cứng – Thực thi chỉ lệnh theo cấu trúc dòng chảy (xem hình 7.9 trong chương sau) – Một lệnh thực thi trong 1 chu kì xung nhịp	– Sử dụng rất nhiều code trong ROM giải mã các chỉ lệnh – Các máy cũ phải tuần tự hết dòng lệnh này mới đến dòng lệnh khác – Cần nhiều chu kì xung nhịp để hoàn thành một lệnh

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 6

1. Giả sử cần thiết kế máy với ký tự 8 bit và bộ nhớ chính chứa 2^{24} ký tự. Hãy cho biết trường địa chỉ cần bao nhiêu bit trong trường hợp:
   a) Ô nhớ kích thước 8 bit
   b) Ô nhớ kích thước 16 bit
   c) Ô nhớ kích thước 32 bit
2. Thiết kế opcode mở rộng nhằm cho phép mã hóa nội dung sau trong lệnh 36 bit
   - 7 lệnh có hai địa chỉ 15 bit và một số hiệu thanh ghi 3 bit
   - 500 lệnh có một địa chỉ 15 bit và một số hiệu thanh ghi 3 bit
   - 50 lệnh không có địa chỉ hoặc thanh ghi
3. Có thể thiết kế opcode mở rộng để cho phép mã hóa nội dung sau trong lệnh 12 bit được không? Trường thanh ghi rộng 3 bit.
   - 4 lệnh có ba thanh ghi
   - 255 lệnh có hai thanh ghi
   - 2048 lệnh không có thanh ghi

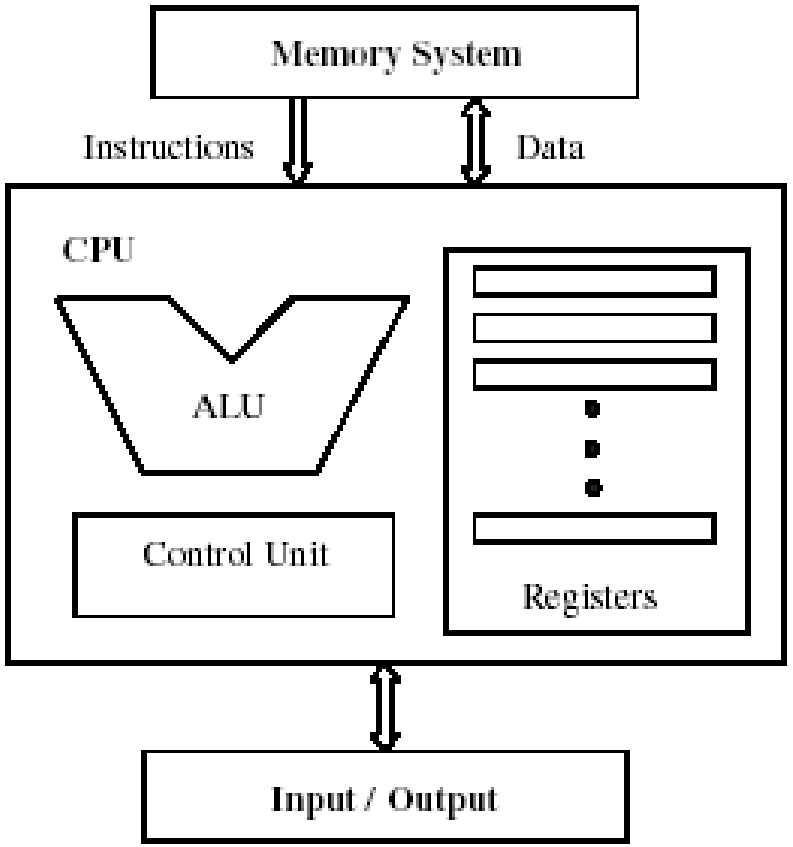
Chương 7
Tổ chức bộ xử lý

Nội dung

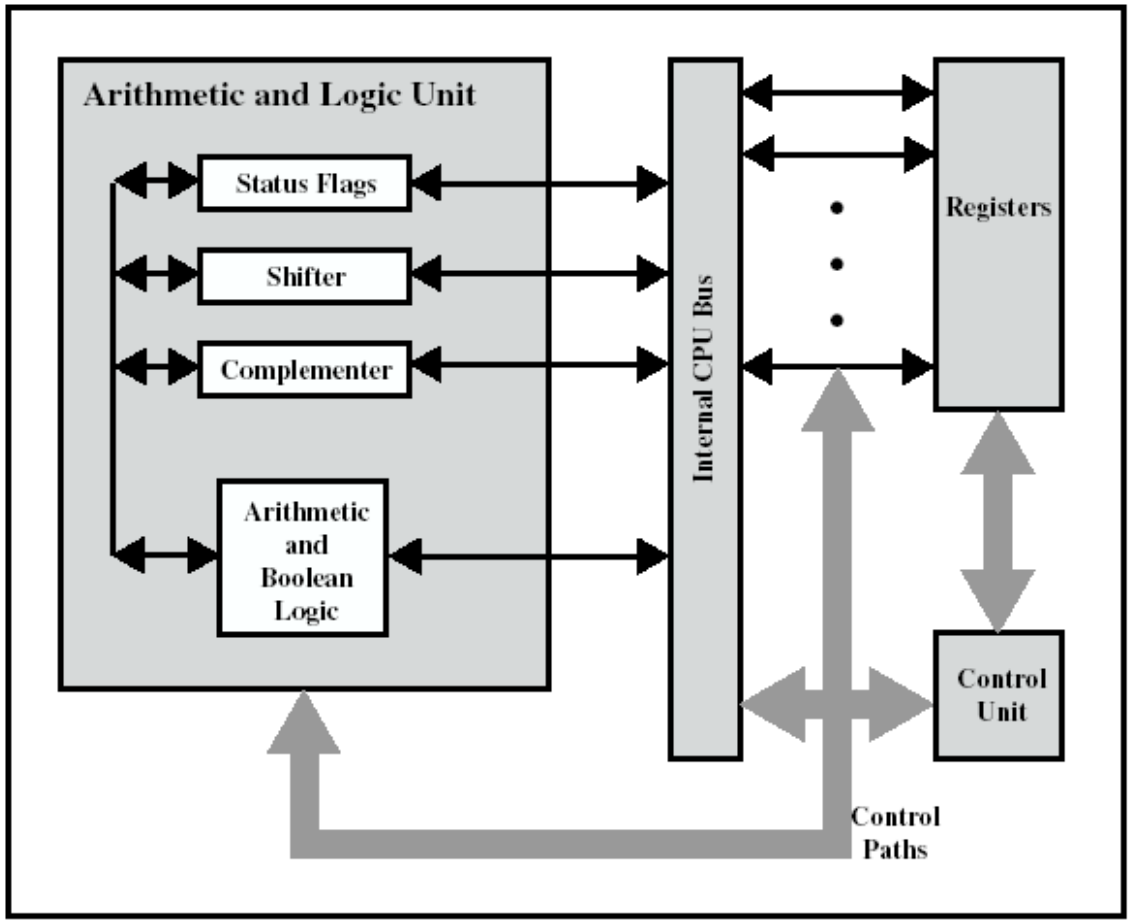
7.1 Tổ chức bộ xử lý trung tâm
7.2 Bộ thanh ghi
7.3 Đường đi dữ liệu (Datapath)
    Tổ chức One-Bus
    Tổ chức Two-Bus, Three-Bus
7.4 Diễn tiến thi hành lệnh mã máy
7.5 Bộ điều khiển
7.6 Xử lý ngắt (Interrupt Handling)
7.7 Kỹ thuật ống dẫn (Pipeline)

7.1. Tổ chức bộ xử lý trung tâm

- Đòi hỏi ở bên trong CPU:
  - Tìm nạp lệnh (Fetch Instruction)
  - Diễn giải lệnh (Interpret Instruction)
  - Tìm nạp dữ liệu (Fetch data)
  - Xử lý dữ liệu (Process data)
  - Ghi dữ liệu (Write data)



Cấu trúc bên trong của CPU

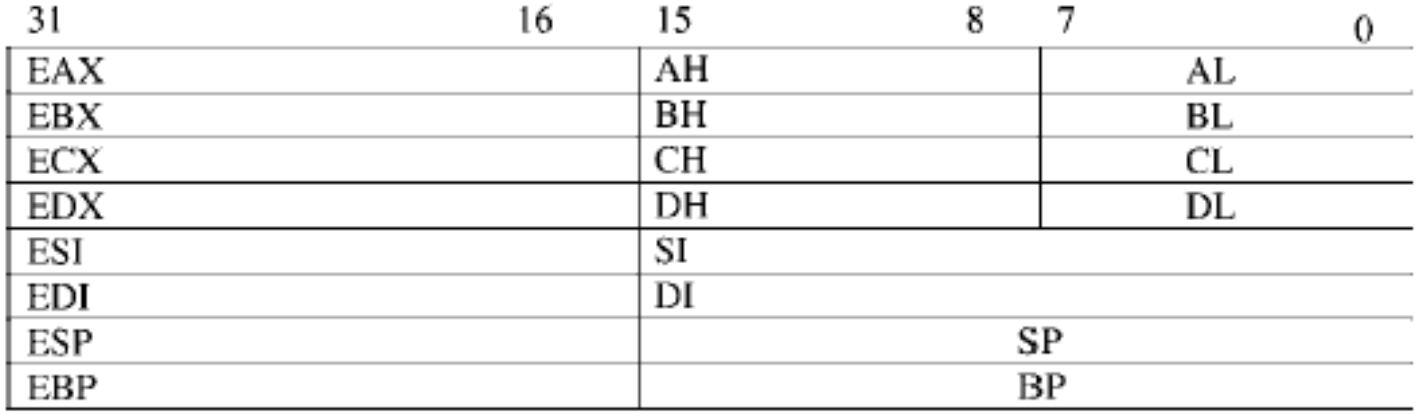


7.2. Bộ thanh ghi

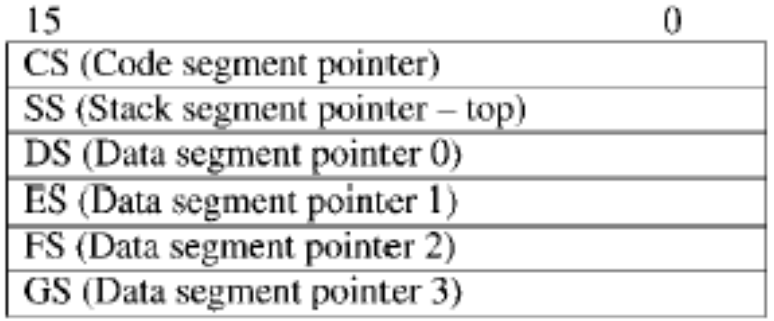
- Thanh ghi mục đích chung
- Thanh ghi có mục đích đặc biệt
- Chiều dài của thanh ghi
- Số lượng thanh ghi
- ***Thanh ghi truy cập bộ nhớ***
  - Thanh ghi dữ liệu bộ nhớ (*memory data register - MDR*)
  - Thanh ghi địa chỉ bộ nhớ (*memory address regiater – MAR*)
- ***Thanh ghi chuyển tải lệnh***
  - Bộ đếm chương trình (*program counter – PC*)
  - Thanh ghi lệnh (*instruction register – IR*)
- ***Thanh ghi từ trạng thái của chương trình (program status word – PSW).***

Các thanh ghi họ 80x86

- 8 thanh ghi mục đích chung:
  - SI (source index)
  - DI (destination index)
  - SP (stack pointer)
  - BP (base pointer)
- Thanh ghi segment
- Thanh ghi đếm chương trình PC,
- thanh ghi lệnh IR
- thanh ghi cờ trạng thái



General Purpose Registers



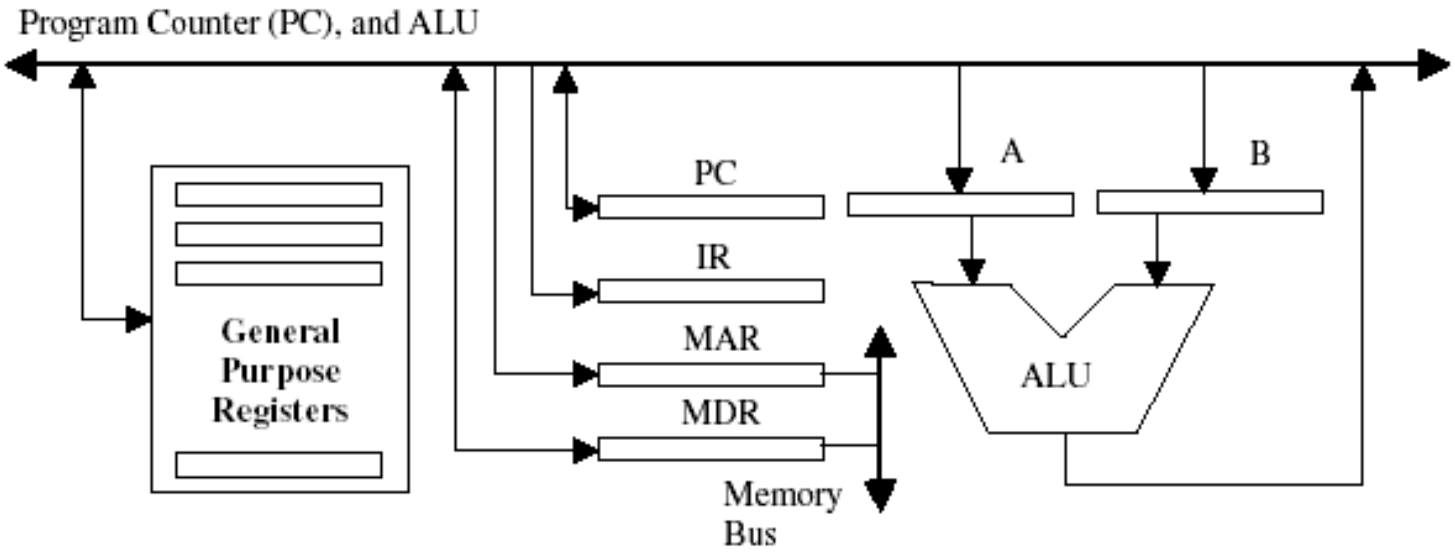
Segment Registers

Flags

7.3. Đường đi dữ liệu (Datapath)

- Đường đi dữ liệu gồm có
  - bộ logic-số học (ALU: Arithmetic and Logic Unit),
  - các mạch dịch,
  - các thanh ghi
  - các đường nối kết các bộ phận trên
- Nhiệm vụ chính của phần đường đi dữ liệu
  - đọc các toán hạng từ các thanh ghi tổng quát
  - thực hiện các phép tính trên toán hạng này trong ALU
  - lưu trữ kết quả trong các thanh ghi tổng quát

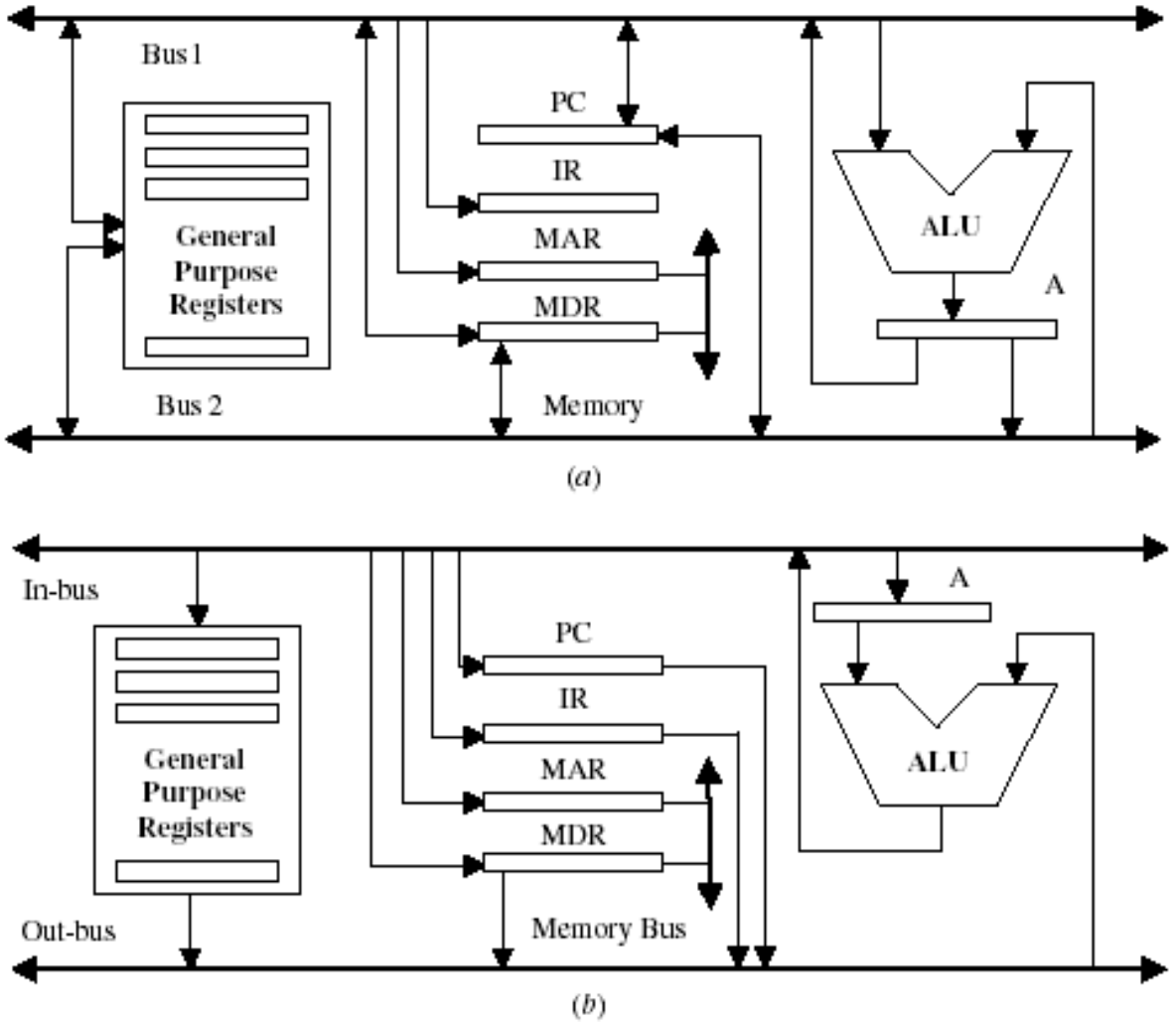
Tổ chức One-Bus



Một BUS chỉ có thể sử dụng một dữ liệu di chuyển trong một chu kỳ đồng hồ

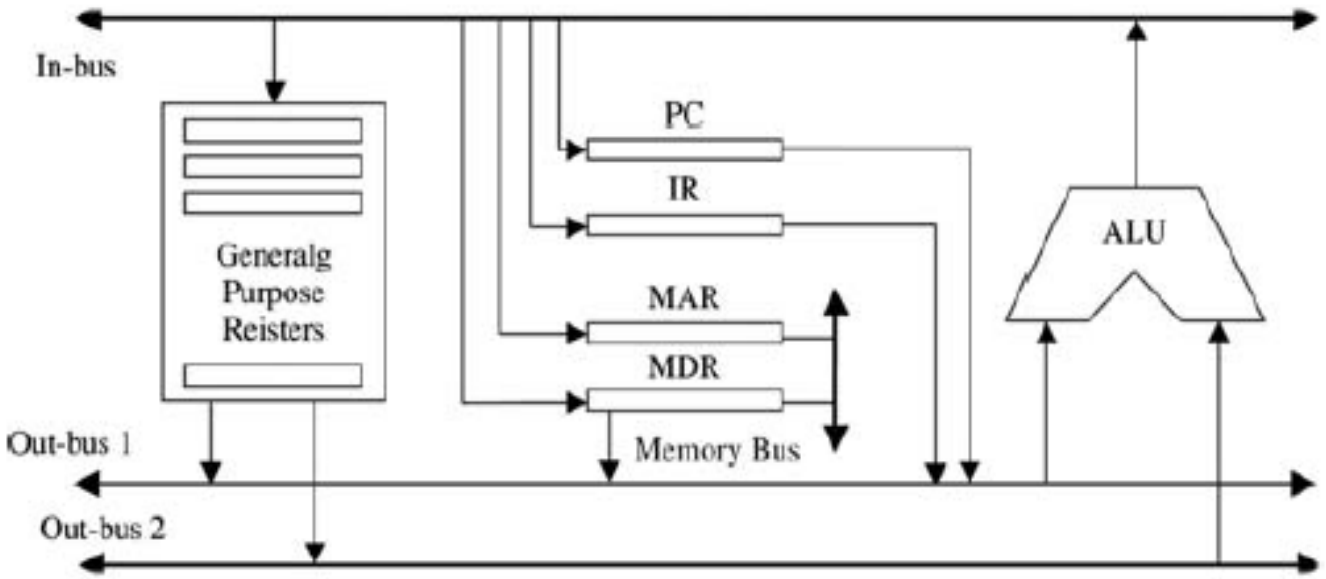
=> một phép toán có hai toán hạng cần hai chu kỳ đồng hồ

Tổ chức *Two-Bus*



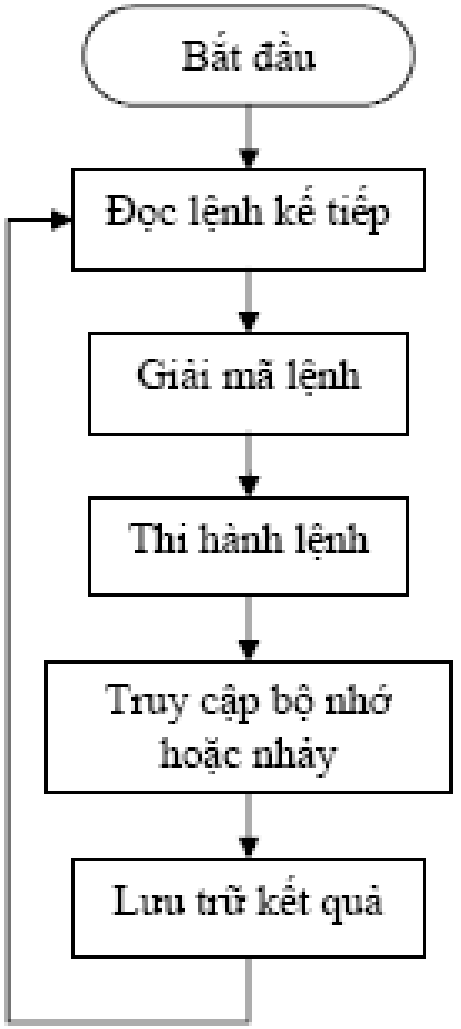
Three-Bus

- Tổ chức đường truyền dữ liệu dạng *three-bus*



7.4. Diễn tiến thi hành lệnh mã máy(*CPU instruction cycle*)

- Việc thi hành một lệnh mã máy có thể chia thành 5 giai đoạn
  - Đọc lệnh (IF: Instruction Fetch)
  - Giải mã lệnh (ID: Instruction Decode)
  - Thi hành lệnh (EX: Execute)
  - Thâm nhập bộ nhớ trong hoặc nhảy (MEM: Memory access)
  - Lưu trữ kết quả (RS: Result Storing).



Đọc lệnh

- ***Đọc lệnh (fetch instruction):***
  - Dữ liệu trong PC được load vào MAR: MAR ← PC
  - Giá trị trong thanh ghi PC tăng lên 1: PC ← PC+1
  - Kết quả của lệnh đọc từ bộ nhớ, dữ liệu được load vào MDR: MDR←M[MAR]
  - Dữ liệu trong MDR được load vào IR: IR ← MDR
- Thứ tự thực hiện lệnh theo thời gian đối với loại *one-bus*:

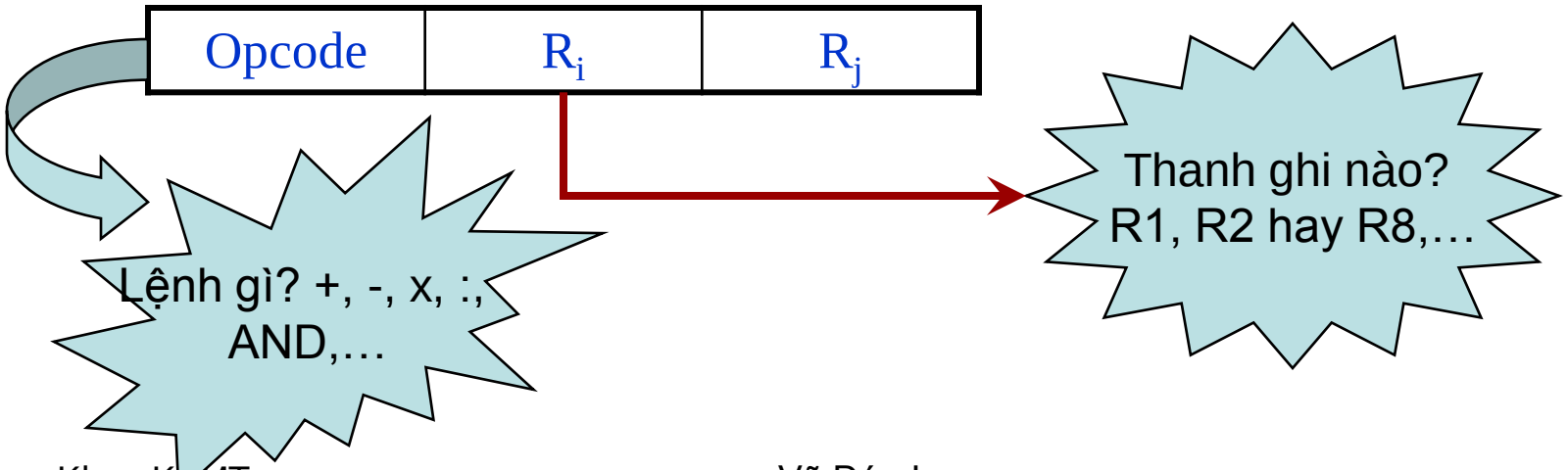
Step	Micro-operation
t_0	MAR ← (PC); A ← (PC)
t_1	MDR ← Mem[MAR]; PC ← (A) + 4
t_2	IR ← (MDR)

Đọc lệnh - Giải mã lệnh

- Thứ tự thực hiện lệnh theo thời gian đối với loại *three-bus:*

Step	Micro-operation
t_0	MAR ← (PC); PC ← (PC) + 4
t_1	MDR ← Mem[MAR]
t_2	IR ← (MDR)

- ***Giải mã lệnh và đọc các thanh ghi nguồn:***



Thi hành một lệnh số học đơn giản

- Ví dụ: ADD R1,R2,R0
- Các bước thi hành lệnh:

1. The registers R_0, R_1, R_2, are extracted from the IR.
2. The contents of R_1 and R_2 are passed to the ALU for addition.
3. The output of the ALU is transferred to R_0.

- Trong cấu trúc one-bus và two-bus

Step	Micro-operation
t_0	$A \leftarrow (R_1)$
t_1	$B \leftarrow (R_2)$
t_2	$R_0 \leftarrow (A) + (B)$

Step	Micro-operation
t_0	$A \leftarrow (R_1) + (R_2)$
t_1	$R_0 \leftarrow (A)$

Thi hành một lệnh số học đơn giản

- Ví dụ lệnh: ADD R0,X

1. The memory location X is extracted from IR and loaded into MAR.
2. As a result of memory read operation, the contents of X are loaded into MDR.
3. The contents of MDR are added to the contents of R_0.

- Đối với cấu trúc one-bus

Step	Micro-operation
t_0	MAR $\leftarrow$ X
t_1	MDR $\leftarrow$ Mem[MAR]
t_2	$A \leftarrow (R_0)$
t_3	$B \leftarrow (MDR)$
t_4	$R_0 \leftarrow (A) + (B)$

Two-bus

Step	Micro-operation
t_0	MAR $\leftarrow$ X
t_1	MDR $\leftarrow$ Mem[MAR]
t_2	$A \leftarrow (R_0) + (MDR)$
t_3	$R_0 \leftarrow (A)$

Three-bus

Step	Micro-operation
t_0	MAR $\leftarrow$ X
t_1	MDR $\leftarrow$ Mem[MAR]
t_2	$R_0 \leftarrow R_0 + (MDR)$

7.5. Bộ điều khiển

❑ *Bộ điều khiển mạch điện tử*

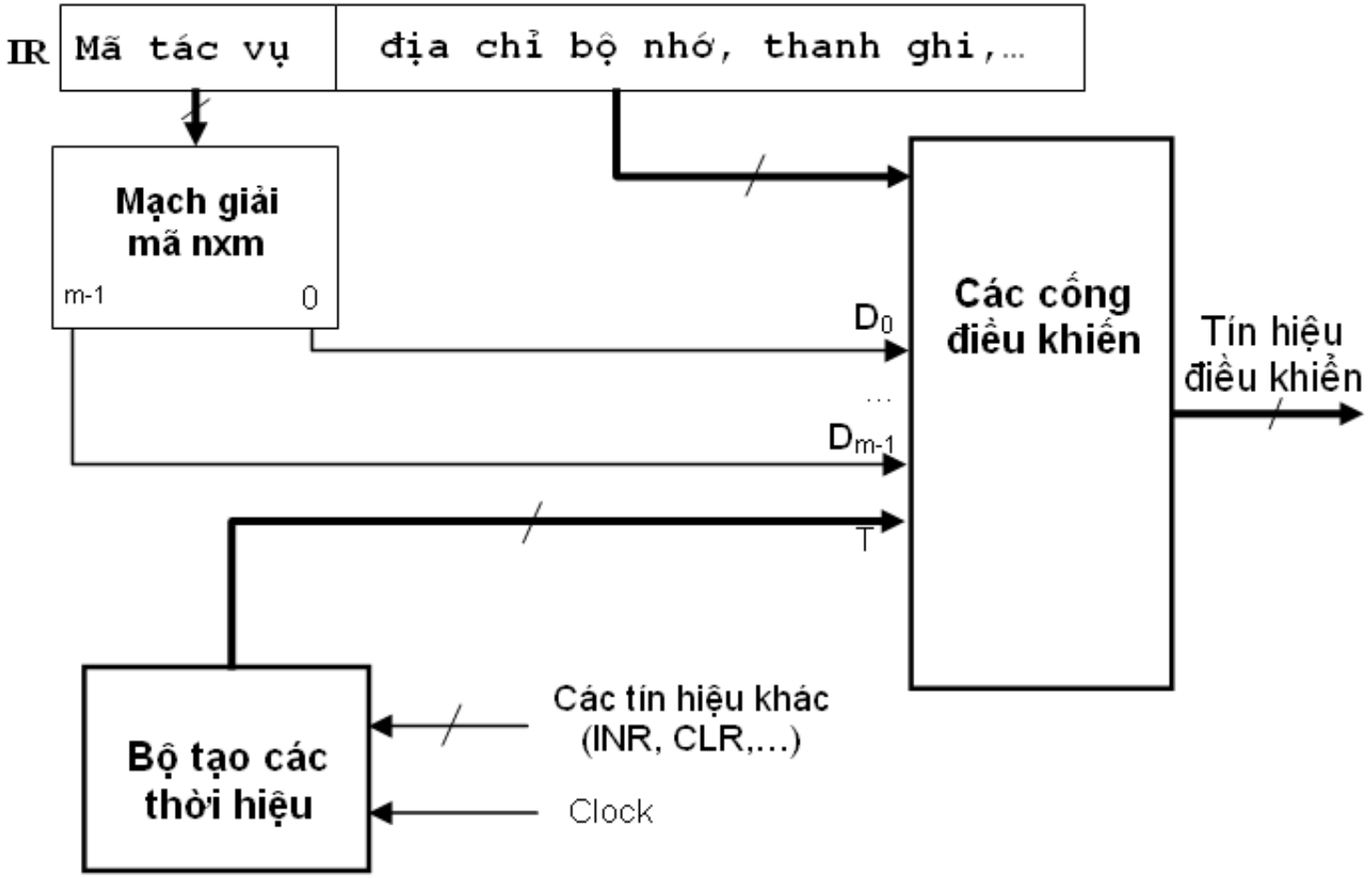
- nguyên lý hoạt động như một mạch tuần tự hay Automate (mạch tự động hóa) trạng thái hữu hạn
- ***Ưu điểm*** :
  - chỉ có một số hữu hạn các trạng thái
  - tối ưu để tạo ra chế độ nhanh cho tác vụ

❑ *Bộ điều khiển vi chương trình*

- dùng một vi chương trình lập sẵn nằm trong bộ nhớ điều khiển (control memory) để khởi động dãy vi tác vụ theo yêu cầu.
- dùng rộng rãi trong các bộ xử lý CISC

Bộ điều khiển (tt)

- sơ đồ khối một bộ điều khiển mạch điện tử cơ bản



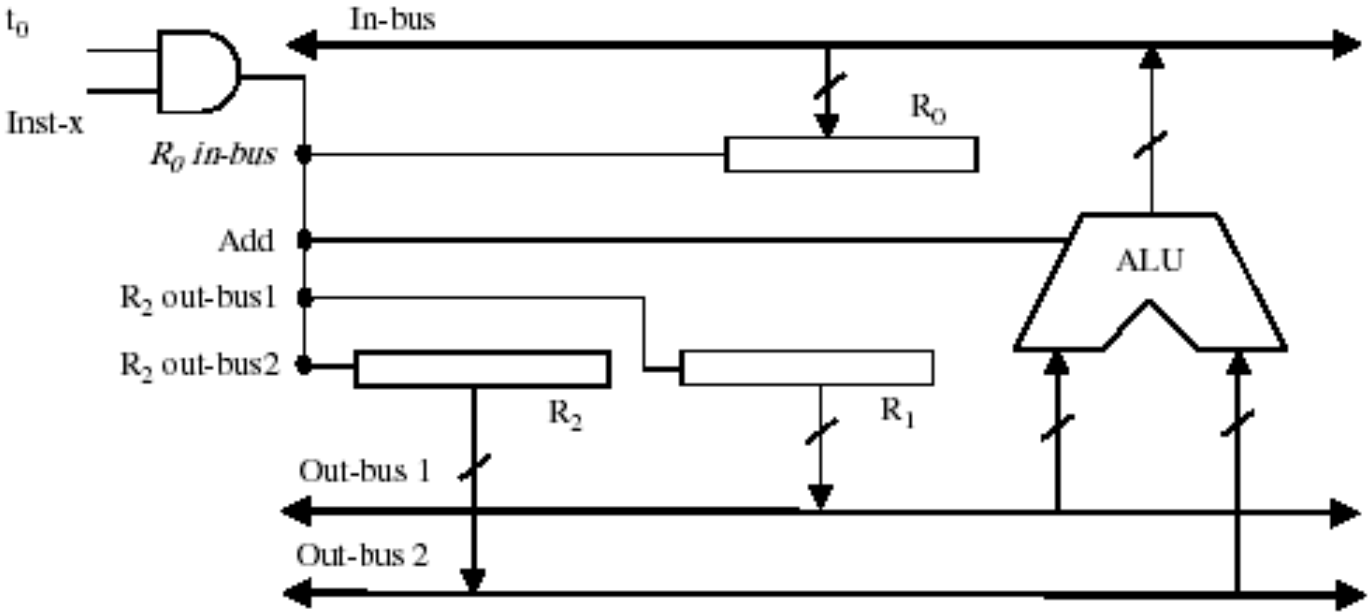
Bộ điều khiển điện tử

- Ví dụ điều khiển thực hiện một lệnh: ADD R0,R1,R2

Các bước thực hiện

Step	Instruction type	Micro-operation	Control
t_0	Inst-x	$R_0 \leftarrow (R_1) + (R_2)$	Select R_1 as source 1 on out-bus1 (R_1 out-bus1) Select R_2 as source 2 on out-bus2 (R_2 out-bus2) Select R_0 as destination on in-bus (R_0 in-bus) Select the ALU function Add (Add)

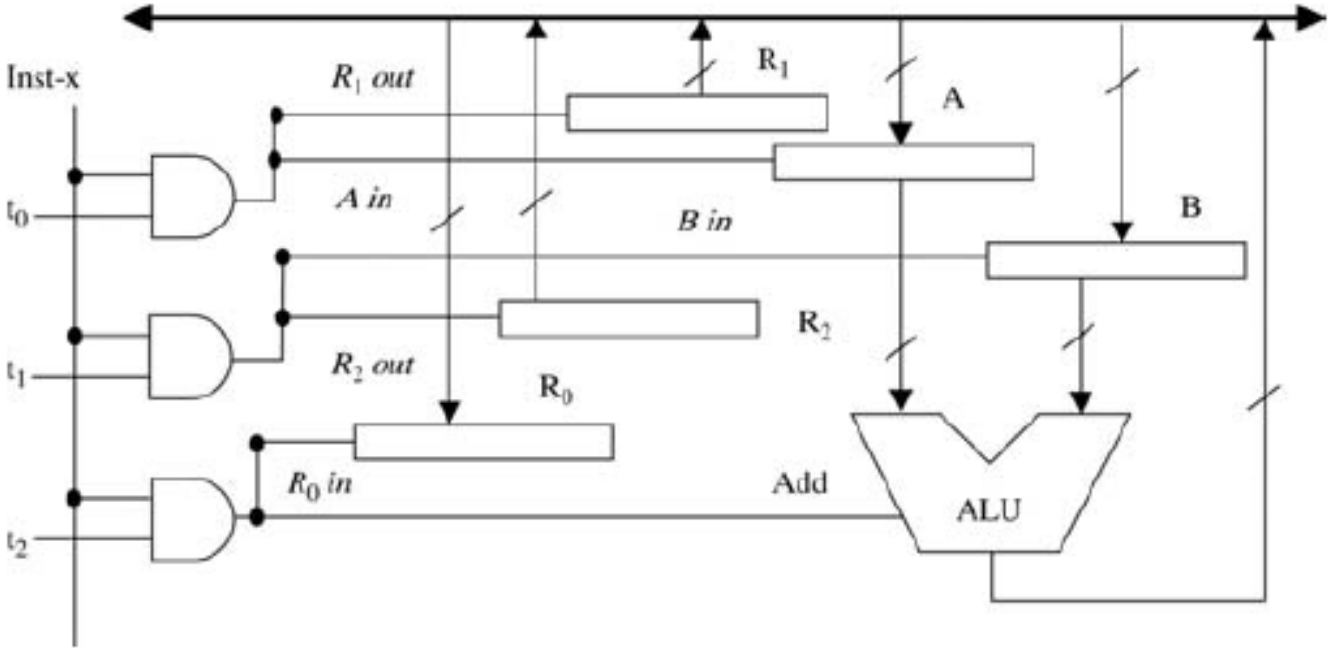
Cài đặt phần cứng



Bộ điều khiển điện tử (tt)

Ví dụ: lệnh ADD R_0, R_1, R_2 với cấu trúc one-bus databath

Step	Instruction type	Micro-operation	
t_0	Inst-x	$A \leftarrow (R_1)$	Select R_1 as source (R_1 out) Select A as destination (A in)
t_1	Inst-x	$B \leftarrow (R_2)$	Select R_2 as source (R_2 out) Select B as destination (B in)
t_2	Inst-x	$R_0 \leftarrow (A) + (B)$	Select the ALU function Add (Add) Select R_0 as destination (R_0 in)



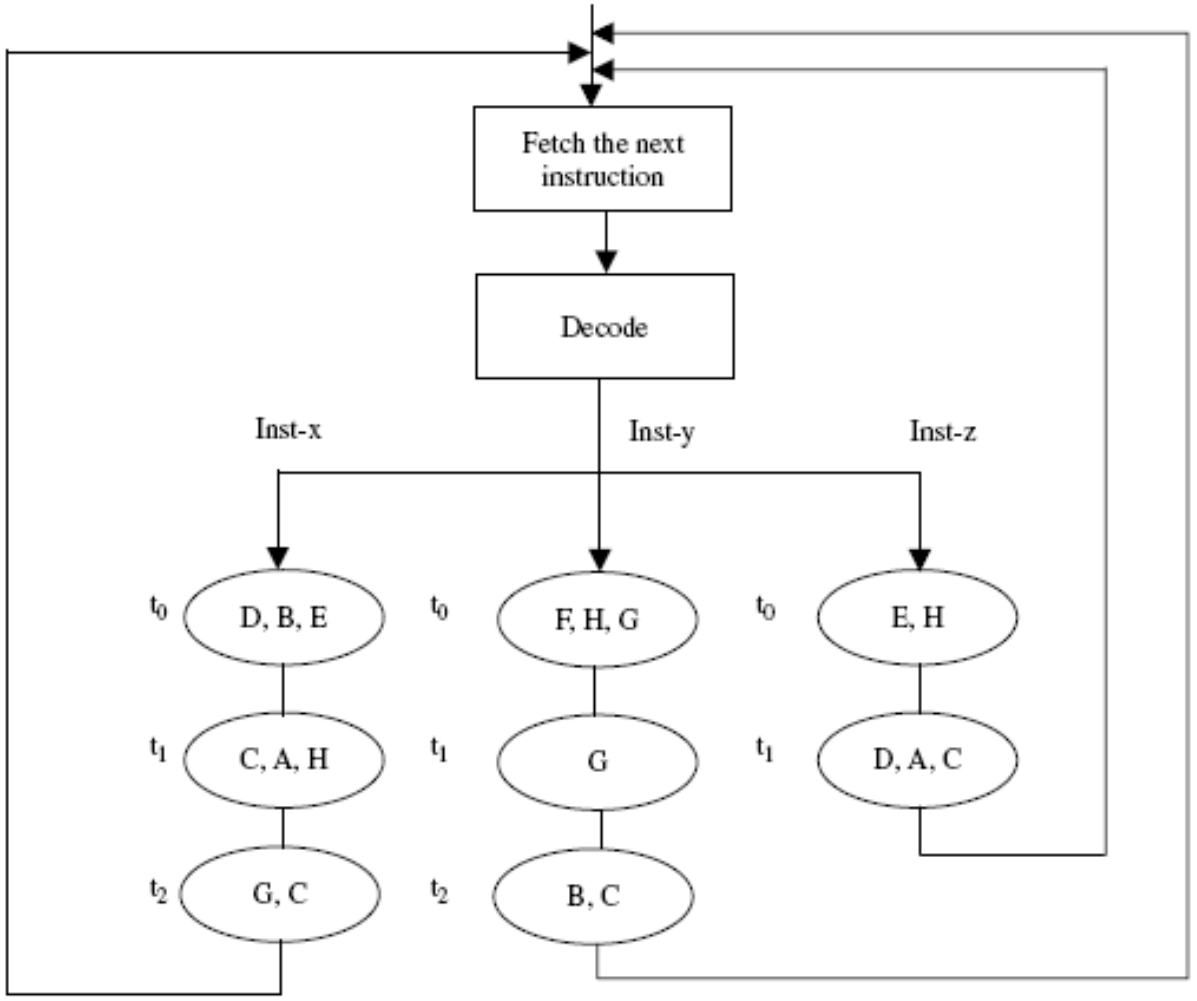
Bộ điều khiển điện tử (tt)

- Yêu cầu tìm ra biểu thức logic cho tín hiệu điều khiển
- Ví dụ: giả sử bộ lệnh có 3 lệnh Inst-x, Inst-y, Inst-z và A,B,C,D,R,F,G,H là các đường điều khiển.

Step	Inst-x	Inst-y	Inst-z
t_0	D, B, E	F, H, G	E, H
t_1	C, A, H	G	D, A, C
t_2	G, C	B, C	

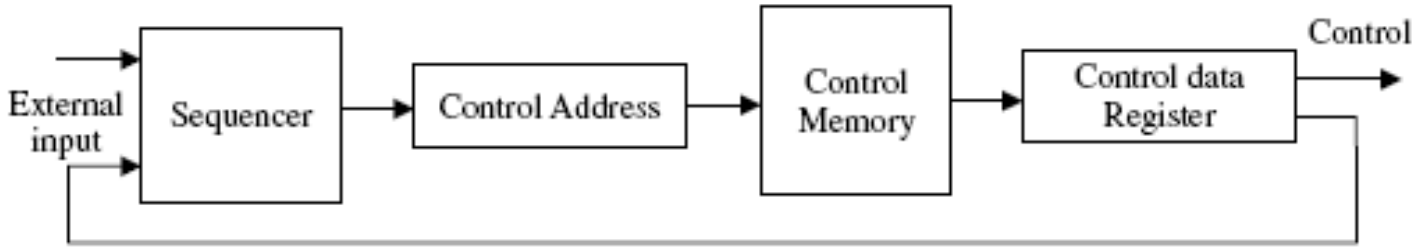
Bộ điều khiển điện tử (tt)

Sơ đồ trạng thài thực hiện lệnh



Bộ điều khiển vi chương trình

- Vi lệnh chiều ngang và chiều dọc



- Ví dụ: Trong cấu trúc đường truyền three-bus, giả sử có 16 thanh ghi GPR R_0-R_{15}, ALU có 8 phép toán (add, sub, multiply, divide, AND,OR, shift left và shift right). Xét format cho vi lệnh ADD R0,R1,R2 theo chiều ngang

Vi lệnh chiều ngang và chiều dọc

Purpose	Number of bits	Explanations
ALU	8 bits	8 functions
Source 1	20 bits	16 general-purpose registers + 4 special-purpose registers
Source 2	16 bits	16 general-purpose registers
Destination	20 bits	16 general-purpose registers + 4 special-purpose registers

Purpose	Number of bits	Explanations
ALU	4 bits	8 functions + none
Source 1	5 bits	16 general-purpose registers + 4 special-purpose registers + none
Source 2	5 bits	16 general-purpose registers + none
Destination	5 bits	16 general-purpose registers + 4 special-purpose registers + none

7.6. Xử lý ngắt (Interrupt Handling)

- Ngắt là một sự kiện xảy ra một cách ngẫu nhiên trong máy tính và làm ngưng tính tuần tự của chương trình (nghĩa là tạo ra một lệnh nhảy)
- Ngắt quãng được dùng cho các công việc:
  - Ngoại vi đòi hỏi nhập hoặc xuất số liệu.
  - Người lập trình muốn dùng dịch vụ của hệ điều hành.
  - Cho một chương trình chạy từng lệnh.
  - Làm điểm dừng của một chương trình.
  - Báo tràn số liệu trong tính toán số học.
  - Trang bộ nhớ thực sự không có trong bộ nhớ.
  - Báo vi phạm vùng cấm của bộ nhớ.
  - Báo dùng một lệnh không có trong tập lệnh.
  - Báo phần cứng máy tính bị hư.
  - Báo điện bị cắt.

7.6. Xử lý ngắt (Interrupt Handling)

- Khi một ngắt xảy ra, bộ xử lý thi hành các bước:
  - 1. Thực hiện xong lệnh đang làm.
  - 2. Lưu trữ trạng thái hiện tại.
  - 3. Nhảy đến chương trình phục vụ ngắt
  - 4. Khi chương trình phục vụ chấm dứt, bộ xử lý khôi phục lại trạng thái cũ của nó và tiếp tục thực hiện chương trình mà nó đang thực hiện khi bị ngắt.
- Thực hiện các vi tác vụ khi ngắt

Step			Micro-operation
t_1	MDR	←	(PC)
t_2	MAR	←	address1 (where to save old PC);
	PC	←	address2 (interrupt handling routine)
t_3	Mem[MAR]	←	(MDR)

7.7. Kỹ thuật ống dẫn (PIPELINE)

❑ Thực hiện lệnh trong kỹ thuật pipeline:

Chuỗi lệnh	Chu kỳ xung nhịp								
	1	2	3	4	5	6	7	8	9
Lệnh thứ i	IF	ID	EX	MEM	RS				
Lệnh thứ i+1		IF	ID	EX	MEM	RS			
Lệnh thứ i+2			IF	ID	EX	MEM	RS		
Lệnh thứ i+3				IF	ID	EX	MEM	RS	
Lệnh thứ i+4					IF	ID	EX	MEM	RS

❑ Một số ràng buộc trong pipeline

- Cần phải có một mạch điện tử để thi hành mỗi giai đoạn của lệnh
- Phải có nhiều thanh ghi khác nhau dùng cho các tác vụ đọc và viết
- Cần phải giải mã các lệnh một cách đơn giản
- Cần phải có các bộ làm tính ALU hữu hiệu để có thể thi hành lệnh số học dài nhất

Những khó khăn trong kỹ thuật ống dẫn

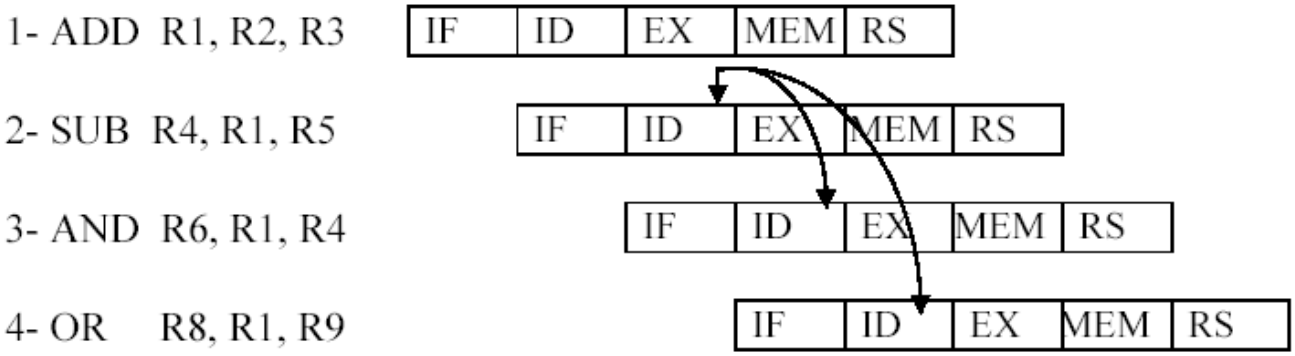
- Khó khăn do cấu trúc
- Khó khăn do điều khiển
- Khó khăn do số liệu
  - Ví dụ trường hợp các lệnh liên tiếp sau:

    Lệnh 1: ADD R1, R2, R3

    Lệnh 2: SUB R4, R1, R5

    Lệnh 3: AND R6, R1, R7

    Lệnh 4: OR R8, R1, R9



Chương 8
Hệ thống bộ nhớ

Nội dung

1. Các cấp bộ nhớ (Memory Hierarchy)
2. Bộ nhớ cache (Cache Memory)
3. Bộ nhớ trong (Main Memory)
4. Bộ nhớ ảo (Virtual Memory)

Các cấp bộ nhớ (Memory Hierarchy)

- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - “RAM”
- External memory
  - Backing store

Memory Hierarchy - Diagram

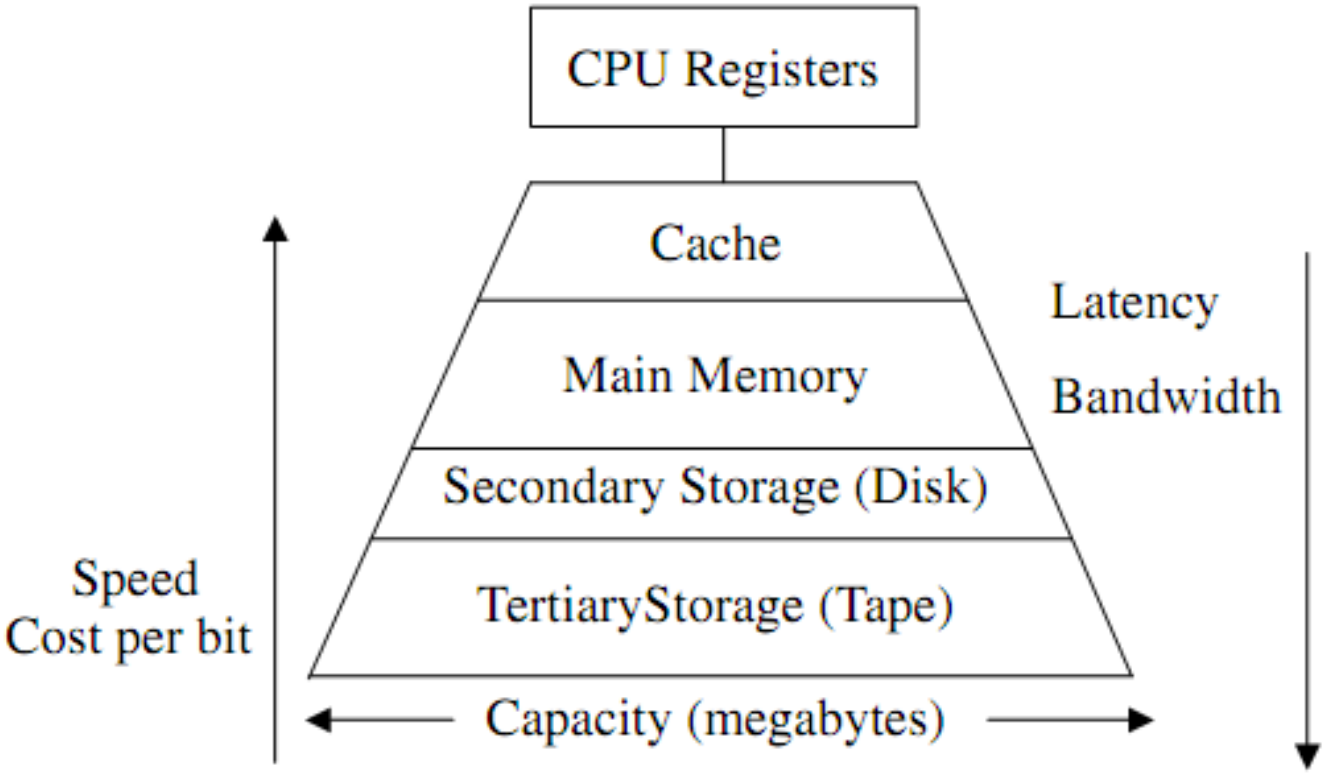


Figure 6.1 Typical memory hierarchy

Performance

- Access time
  - Time between presenting the address and getting the valid data
- Memory Cycle time
  - Time may be required for the memory to “recover” before next access
  - Cycle time is access + recovery
- Transfer Rate
  - Rate at which data can be moved

TABLE 6.1 Memory Hierarchy Parameters

	Access type	Capacity	Latency	Bandwidth	Cost/MB
CPU registers	Random	64–1024 bytes	1–10 ns	System clock rate	High
Cache memory	Random	8–512 KB	15–20 ns	10–20 MB/s	$500
Main memory	Random	16–512 MB	30–50 ns	1–2 MB/s	$20–50
Disk memory	Direct	1–20 GB	10–30 ms	1–2 MB/s	$0.25
Tape memory	Sequential	1–20 TB	30–10,000 ms	1–2 MB/s	$0.025

Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD
- Others
  - Bubble
  - Hologram

Physical Characteristics

- Decay
- Volatility
- Erasable
- Power consumption

Organisation

- Physical arrangement of bits into words
- Not always obvious
- e.g. interleaved

The Bottom Line

- How much?
  - Capacity
- How fast?
  - Time is money
- How expensive?

Hierarchy List

- Registers
- L1 Cache
- L2 Cache
- Main memory
- Disk cache
- Disk
- Optical
- Tape

So you want fast?

- It is possible to build a computer which uses only static RAM (see later)
- This would be very fast
- This would need no cache
  - How can you cache cache?
- This would cost a very large amount

Locality of Reference

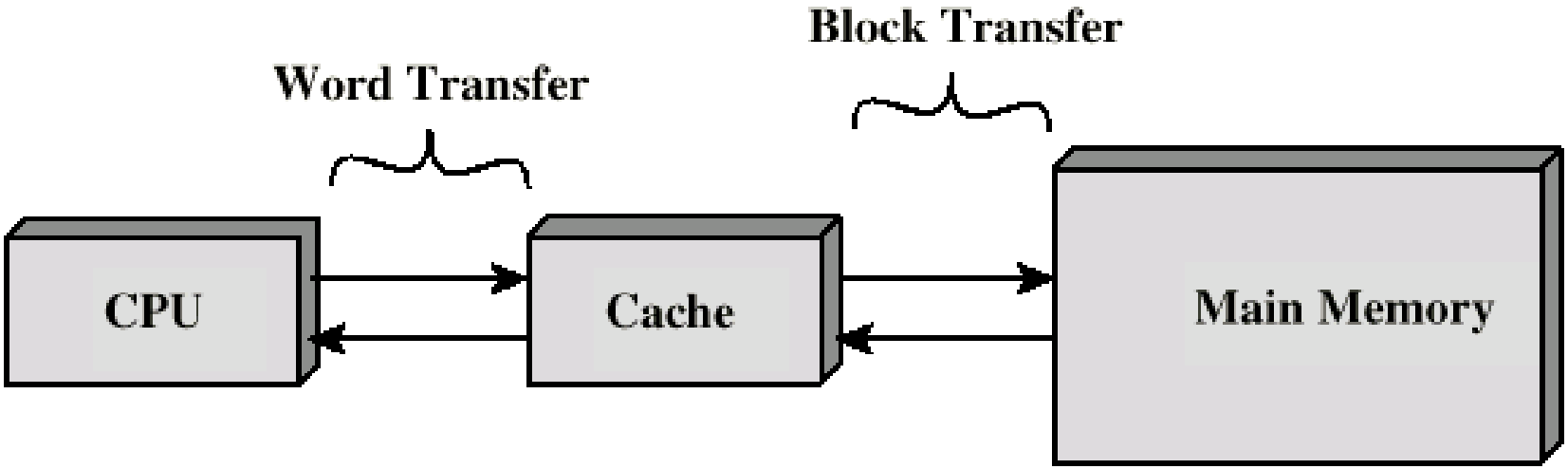
- During the course of the execution of a program, memory references tend to cluster
- e.g. loops

2. Cache

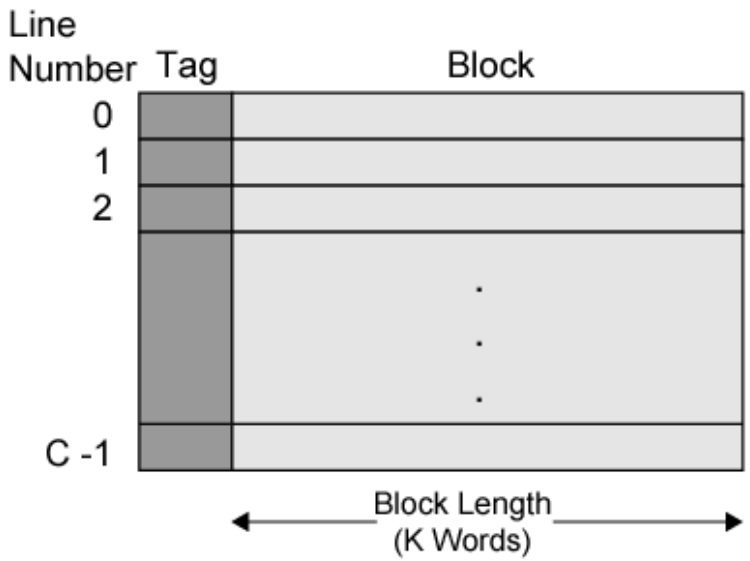
- Tổ ch

Cache

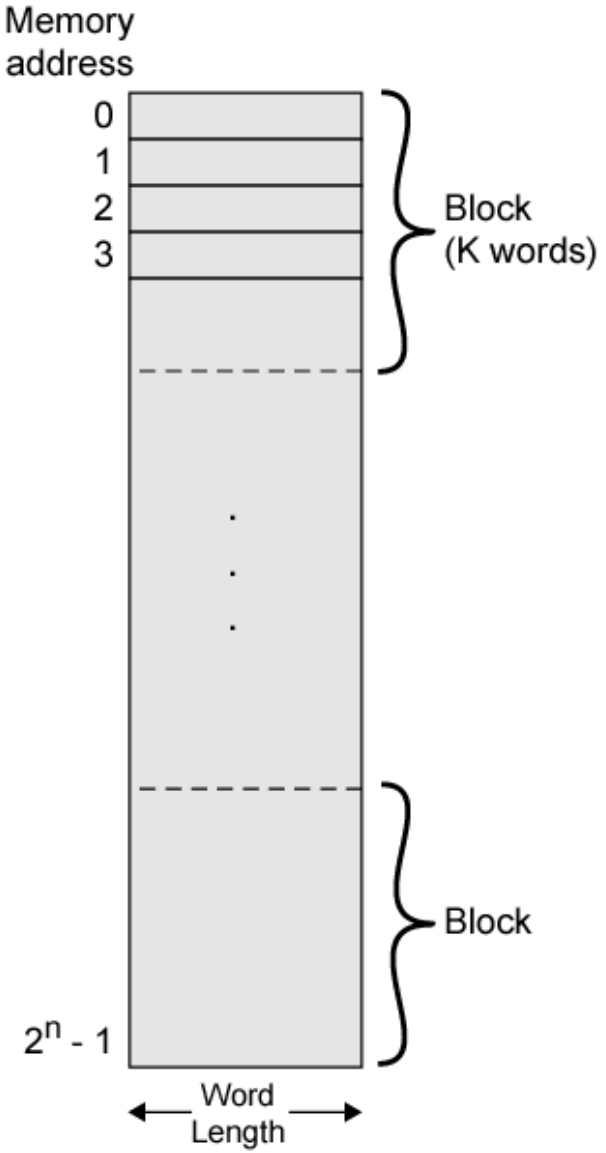
- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module



Cache/Main Memory Structure



(a) Cache

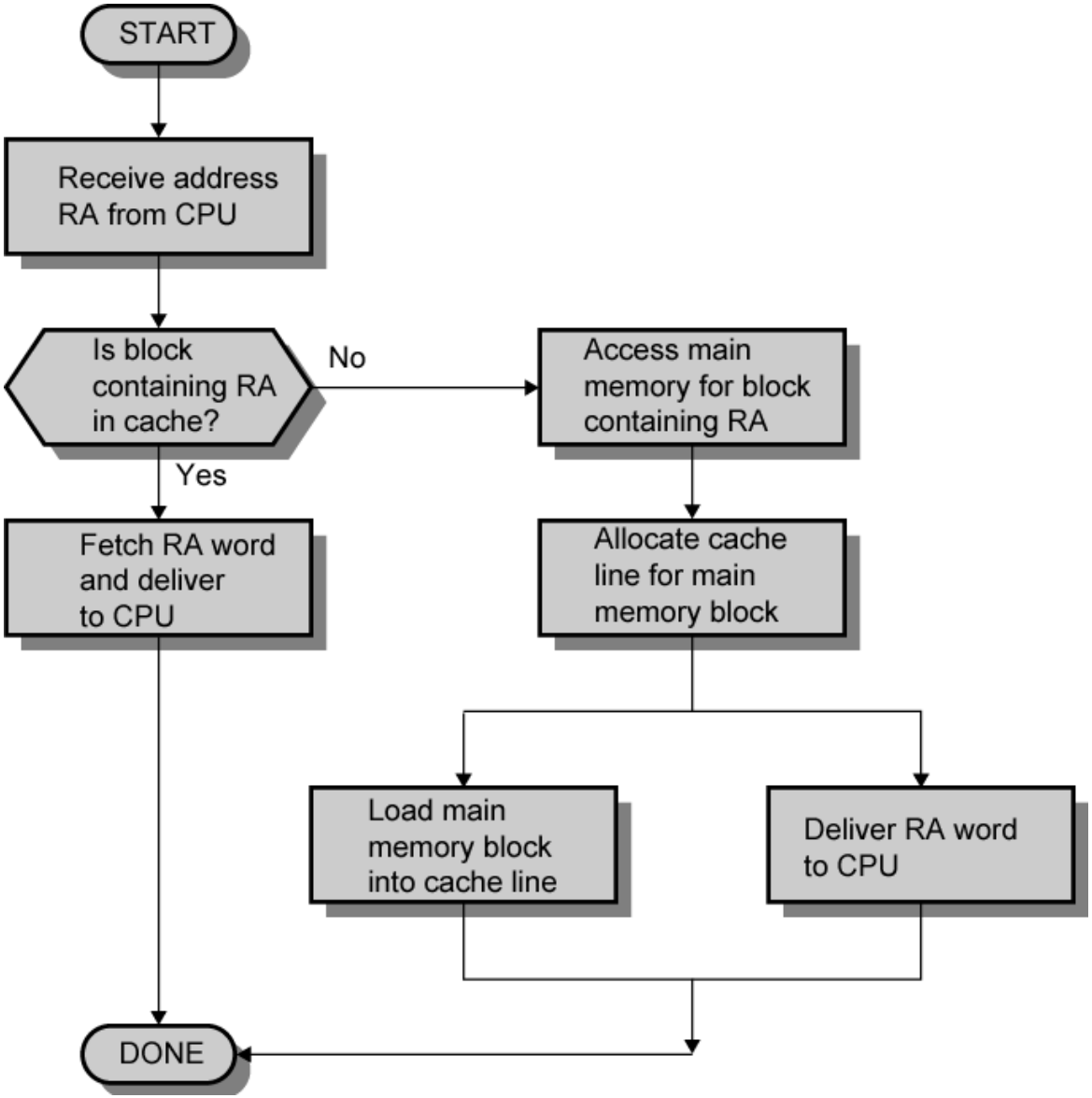


(b) Main memory

Cache operation – overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

Cache Read Operation - Flowchart



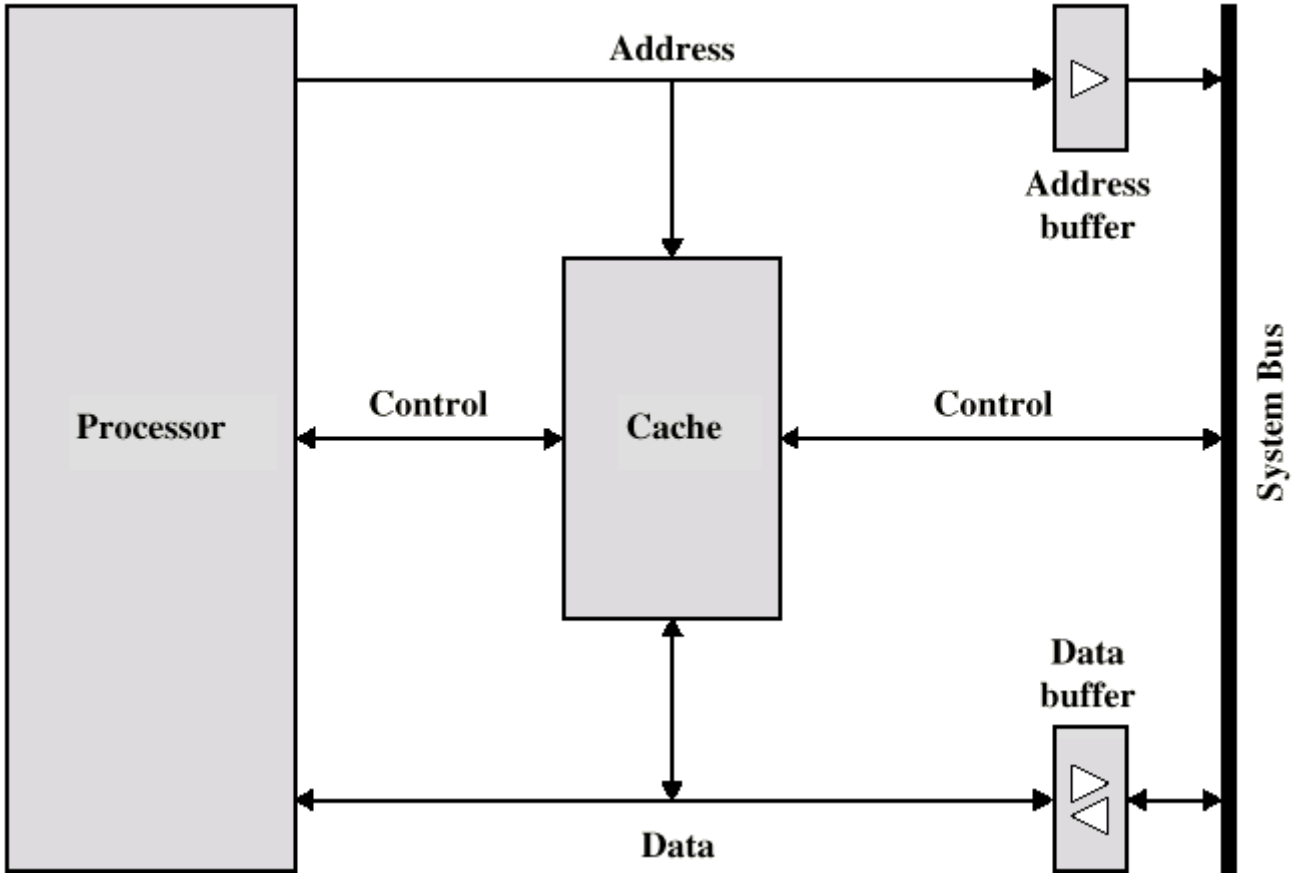
Cache Design

- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

Size does matter

- Cost
  - More cache is expensive
- Speed
  - More cache is faster (up to a point)
  - Checking cache for data takes time

Typical Cache Organization



Comparison of Cache Sizes

Processor	Type	Year of Introduction	L1 cache[a]	L2 cache	L3 cache
IBM 360/85	Mainframe	1968	16 to 32 KB	—	—
PDP-11/70	Minicomputer	1975	1 KB	—	—
VAX 11/780	Minicomputer	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 to 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 to 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 KB	2 MB	—
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 KB/32 KB	4 MB	—
Itanium 2	PC/server	2002	32 KB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 KB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 KB/64 KB	1MB	—

Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

Direct Mapping

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Direct Mapping Address Structure

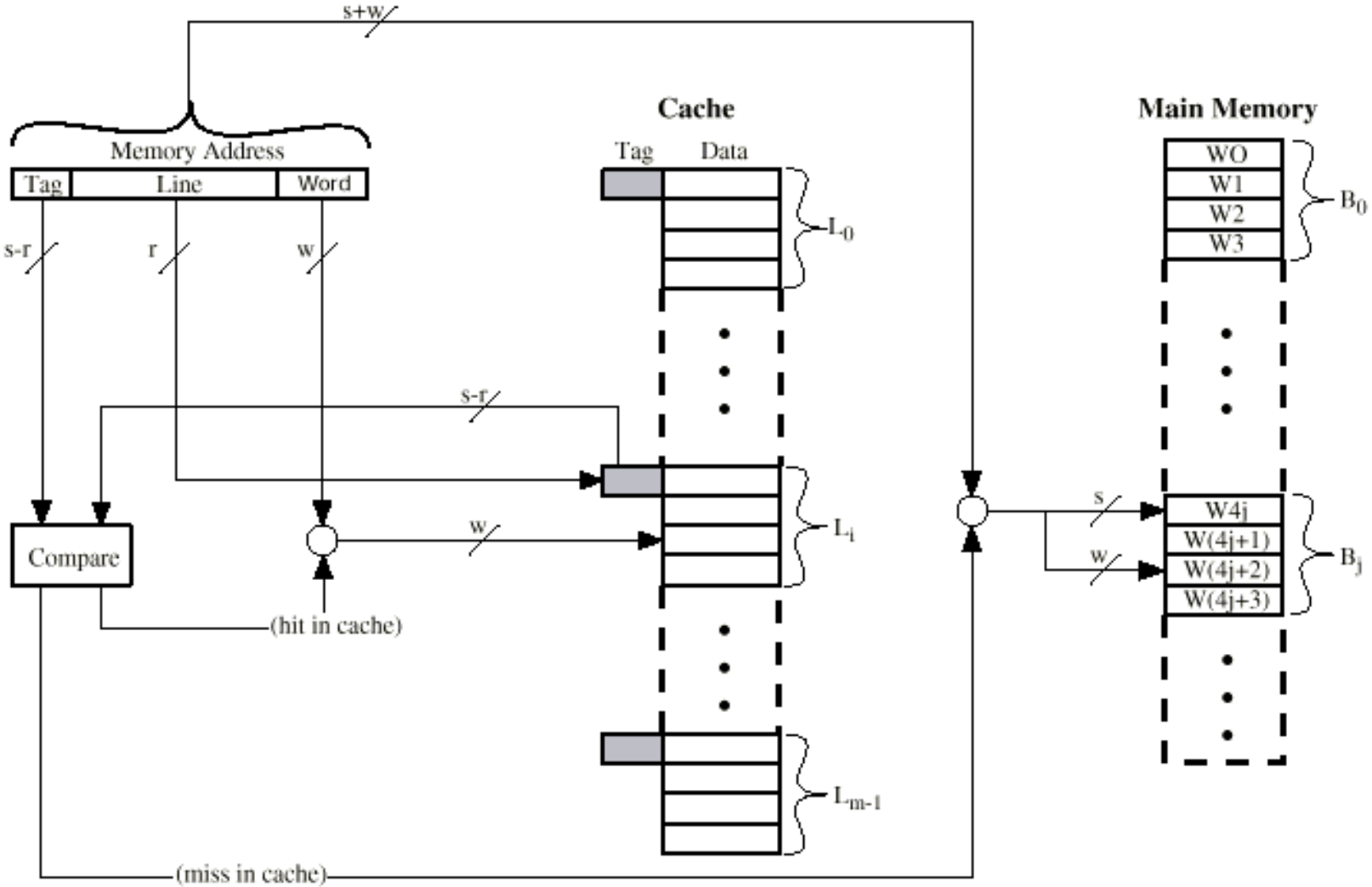
Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

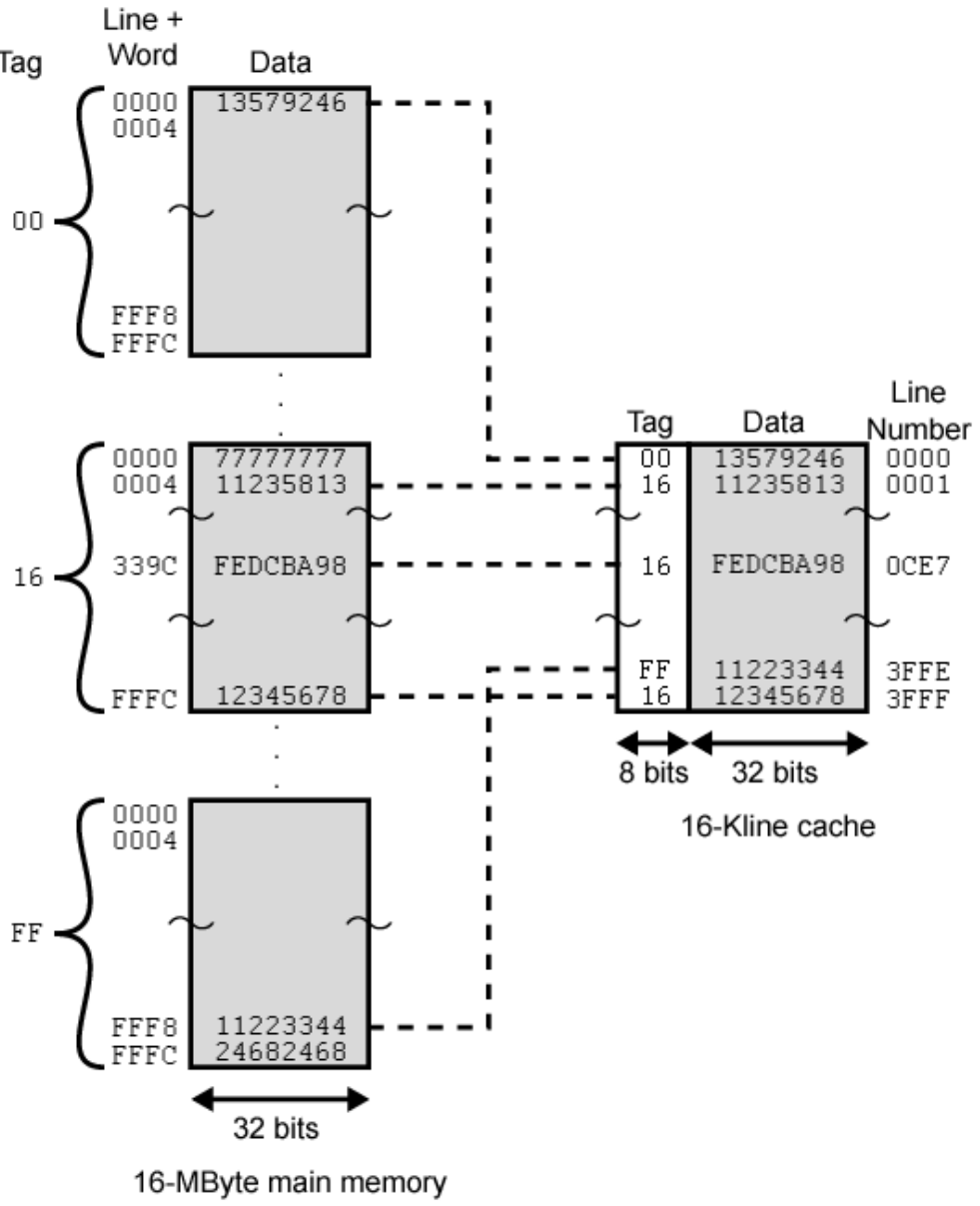
Direct Mapping
Cache Line Table

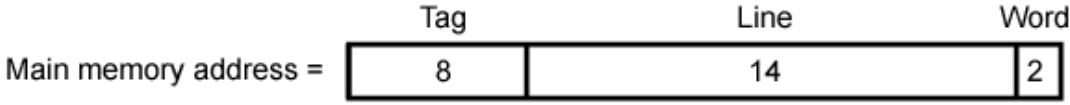
Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2^s-m
1	1,m+1, 2m+1…2^s-m+1
m-1	m-1, 2m-1,3m-1…2^s-1

Direct Mapping Cache Organization



Di


Tag
Line + Word
Data
00
0000
0004
13579246
FFF8
FFFC
16
0000
0004
77777777
11235813
339C
FEDCBA98
FFFC
12345678
FF
0000
0004
11223344
24682468
FFF8
FFFC
32 bits
16-MByte main memory
Tag
Data
Line Number
00
16
13579246
11235813
0000
0001
16
FEDCBA98
0CE7
FF
16
11223344
12345678
3FFE
3FFF
8 bits
32 bits
16-Kline cache


Tag
Line
Word
Main memory address =
8
14
2

Direct Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+ w}/2^{w} = 2^{s}$
- Number of lines in cache = m = 2^{r}
- Size of tag = (s – r) bits

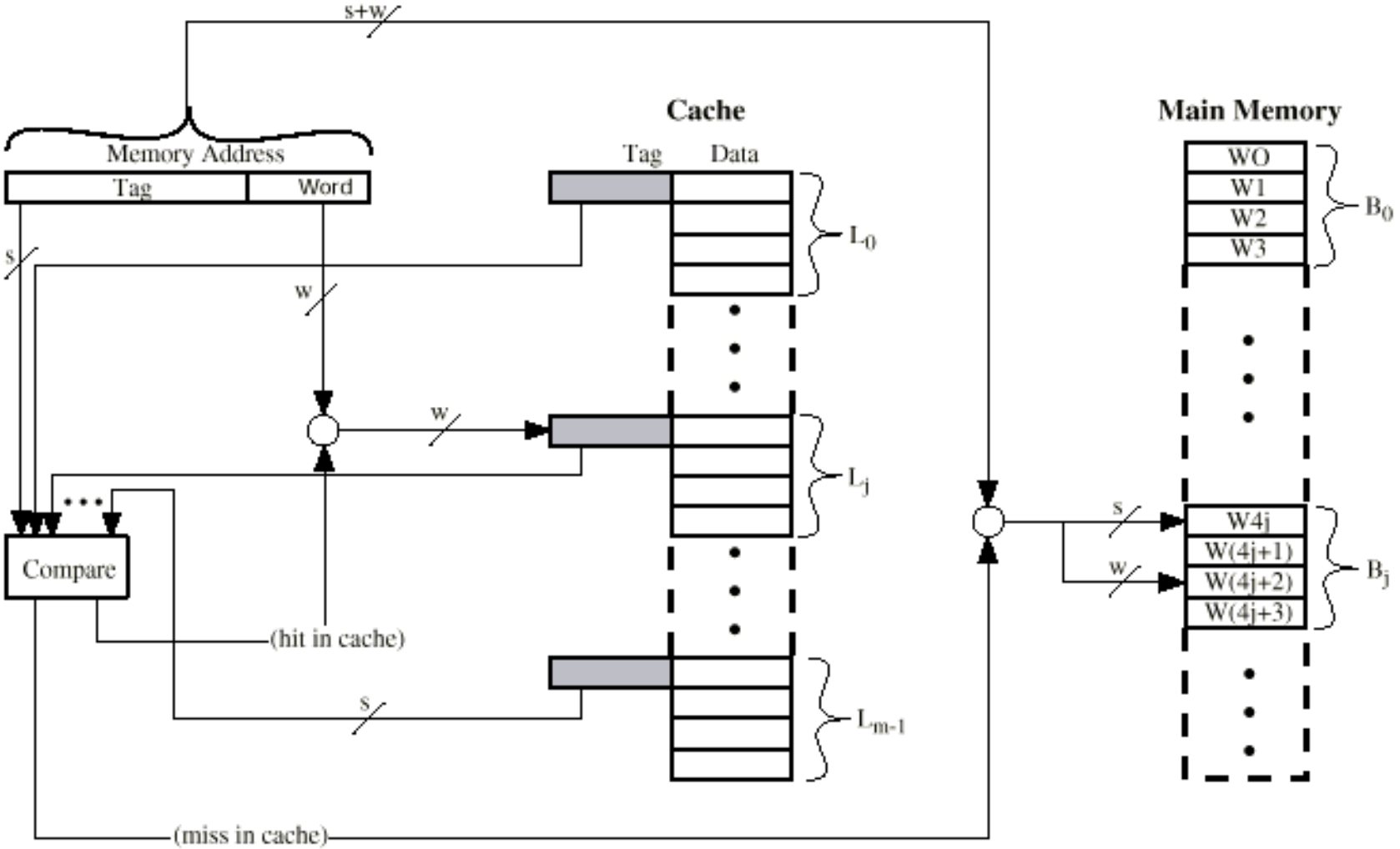
Direct Mapping pros & cons

- Simple
- Inexpensive
- Fixed location for given block
  - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

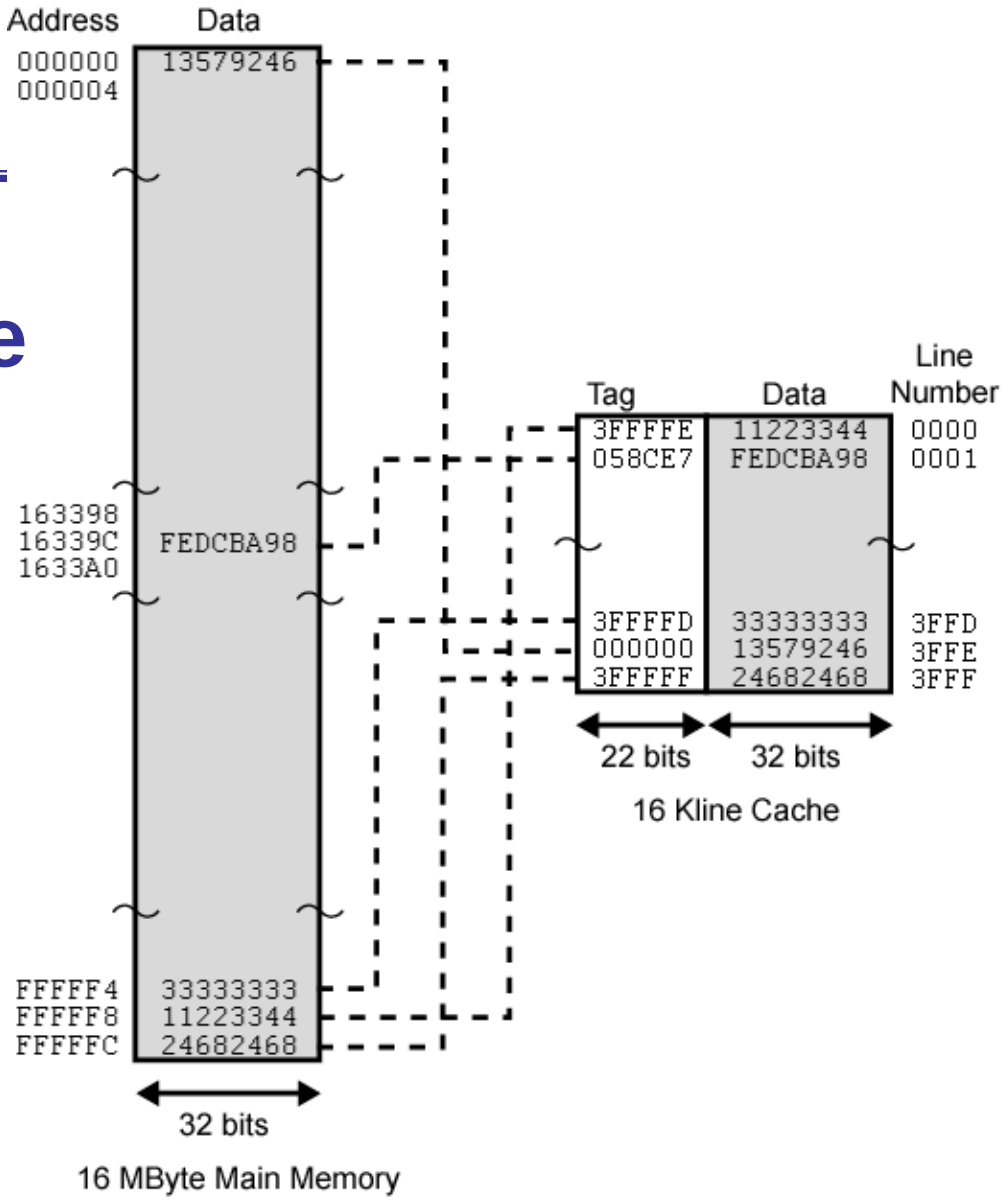
Associative Mapping

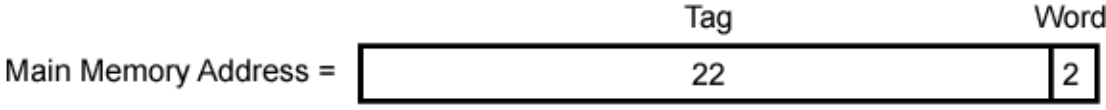
- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

Fully Associative Cache Organization



Associative Mapping Example





Associative Mapping Address Structure

Tag 22 bit	Word 2 bit

- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block
- e.g.

Address	Tag	Data	Cache line
FFFFFC	3FFFFF	24682468	3FFF

Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+ w}/2^{w} = 2^{s}$
- Number of lines in cache = undetermined
- Size of tag = s bits

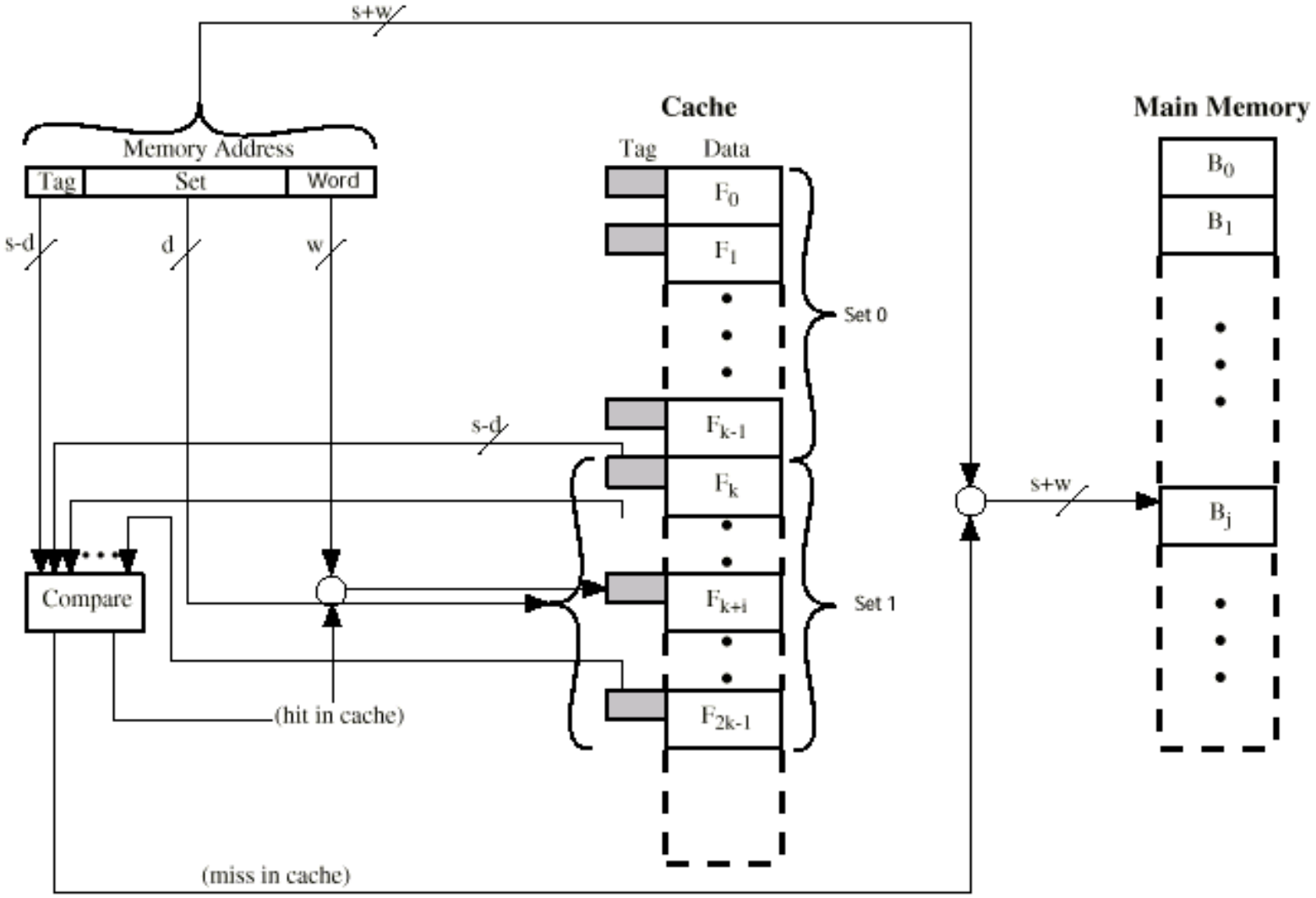
Set Associative Mapping

- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

Set Associative Mapping Example

- 13 bit set number
- Block number in main memory is modulo 2^{13}
- 000000, 00A000, 00B000, 00C000 … map to same set

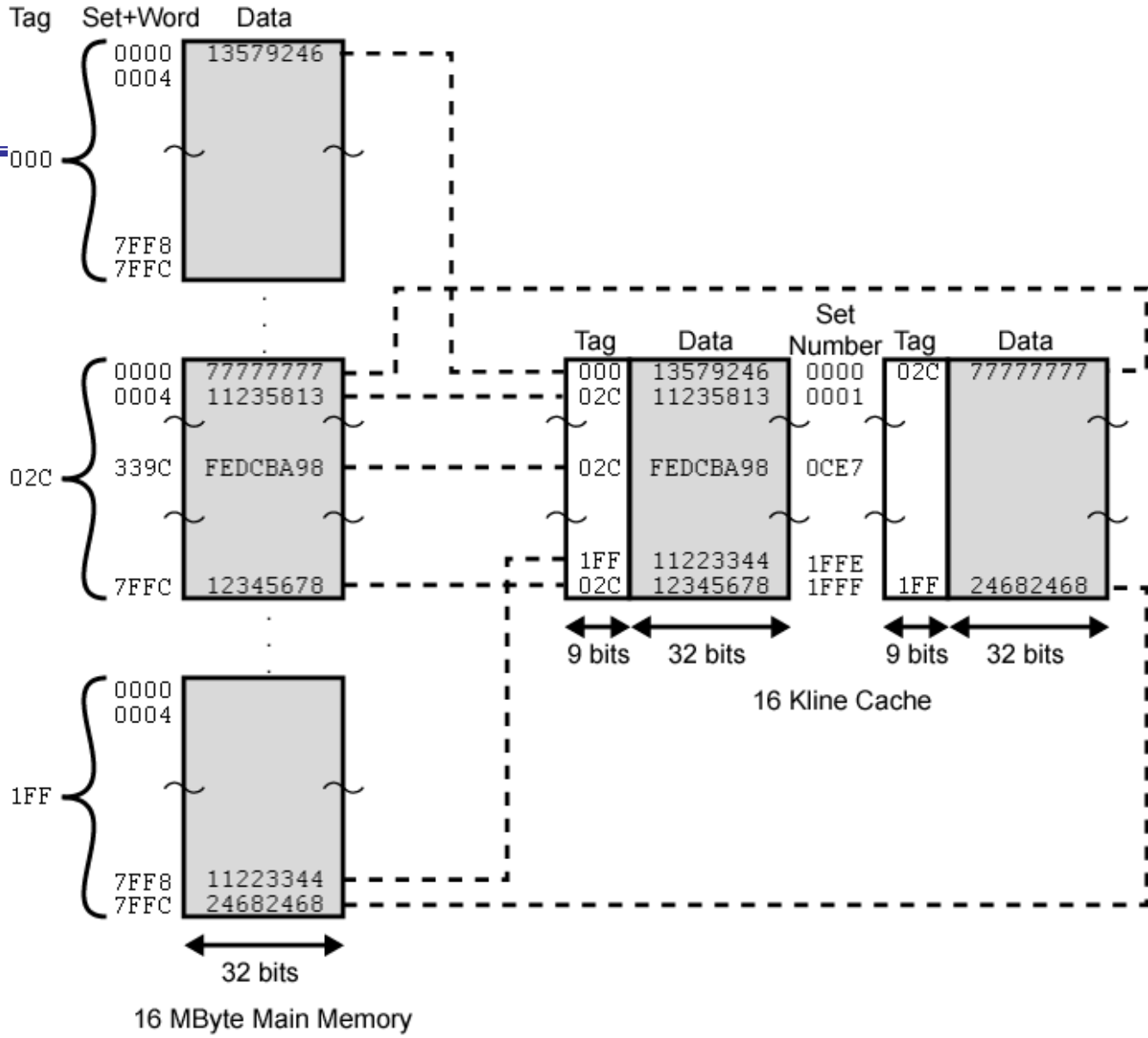
Two Way Set Associative Cache Organization

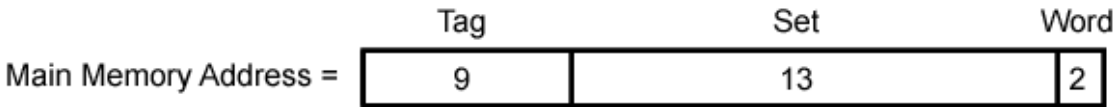


Set Associative Mapping Address Structure

Tag 9 bit	Set 13 bit	Word 2 bit

- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit
- e.g
  - Address Tag Data Set number
  - 1FF 7FFC 1FF 12345678 1FFF
  - 001 7FFC 001 11223344 1FFF


Tag
Set+Word
Data
0000
0004
13579246
000
7FF8
7FFC
0000
0004
77777777
11235813
02C
339C
FEDCBA98
7FFC
12345678
1FF
0000
0004
7FF8
7FFC
11223344
24682468
32 bits
16 MByte Main Memory
Tag
Data
Set Number
Tag
Data
000
02C
13579246
11235813
0000
0001
02C
77777777
02C
FEDCBA98
0CE7
1FF
02C
11223344
12345678
1FFE
1FFF
1FF
24682468
9 bits
32 bits
9 bits
32 bits
16 Kline Cache


Tag
Set
Word
Main Memory Address =
9
13
2

Set Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = 2^{d}
- Number of lines in set = k
- Number of sets = v = 2^{d}
- Number of lines in cache = kv = k * 2^{d}
- Size of tag = (s – d) bits

Replacement Algorithms (1)
Direct mapping

- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms (2) Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
- e.g. in 2 way set associative
  - Which of the 2 block is lru?
- First in first out (FIFO)
  - replace block that has been in cache longest
- Least frequently used
  - replace block which has had fewest hits
- Random

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

- Remember bogus write through caches!

Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

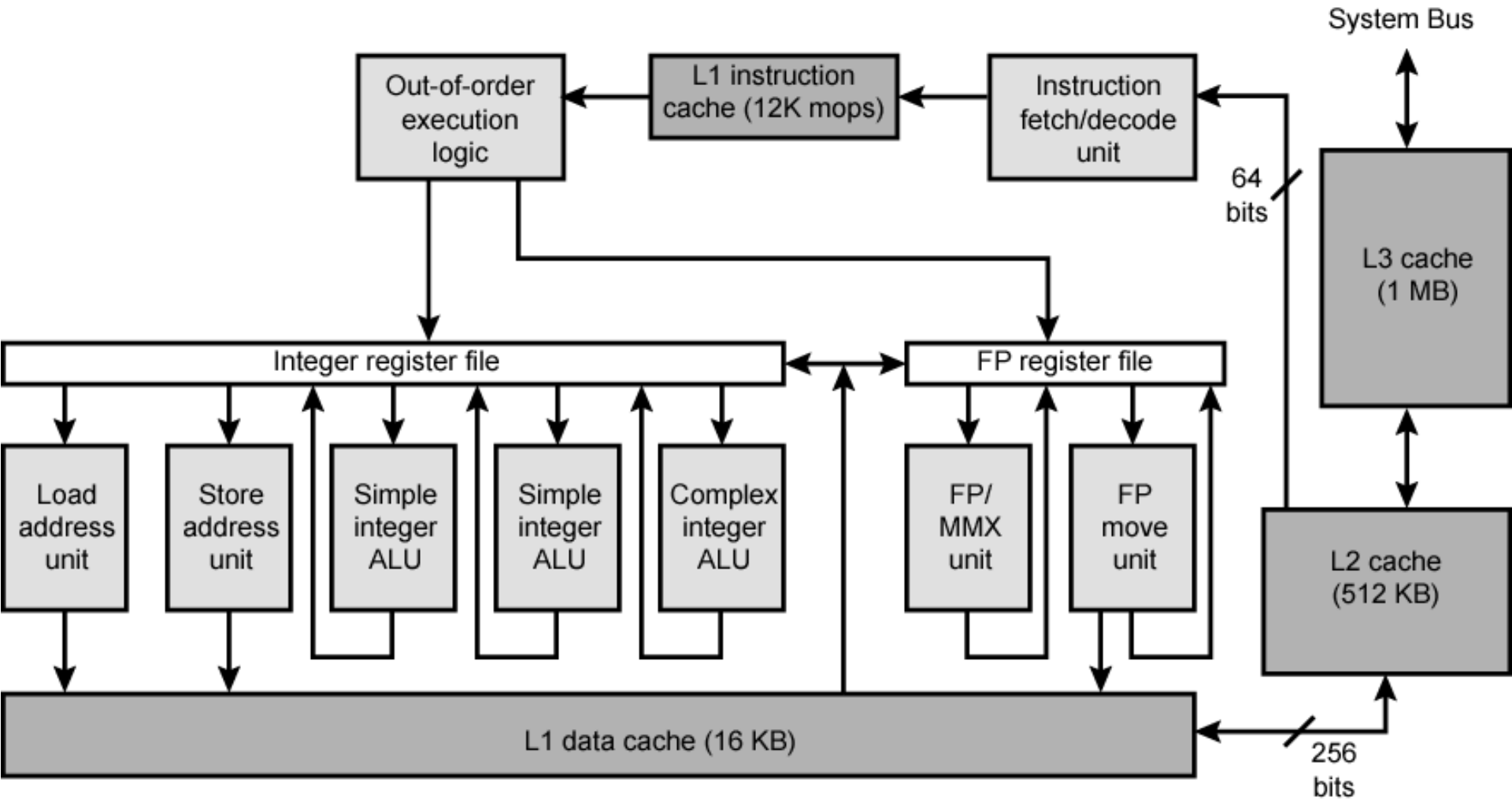
Pentium 4 Cache

- 80386 – no on chip cache
- 80486 – 8k using 16 byte lines and four way set associative organization
- Pentium (all versions) – two on chip L1 caches
  - Data & instructions
- Pentium III – L3 cache added off chip
- Pentium 4
  - L1 caches
    - 8k bytes
    - 64 byte lines
    - four way set associative
  - L2 cache
    - Feeding both L1 caches
    - 256k
    - 128 byte lines
    - 8 way set associative
  - L3 cache on chip

Intel Cache Evolution

Problem	Solution	Processor on which feature first appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Pentium 4 Block Diagram



Pentium 4 Core Processor

- Fetch/Decode Unit
  - Fetches instructions from L2 cache
  - Decode into micro-ops
  - Store micro-ops in L1 cache
- Out of order execution logic
  - Schedules micro-ops
  - Based on data dependence and resources
  - May speculatively execute
- Execution units
  - Execute micro-ops
  - Data from L1 cache
  - Results in registers
- Memory subsystem
  - L2 cache and systems bus

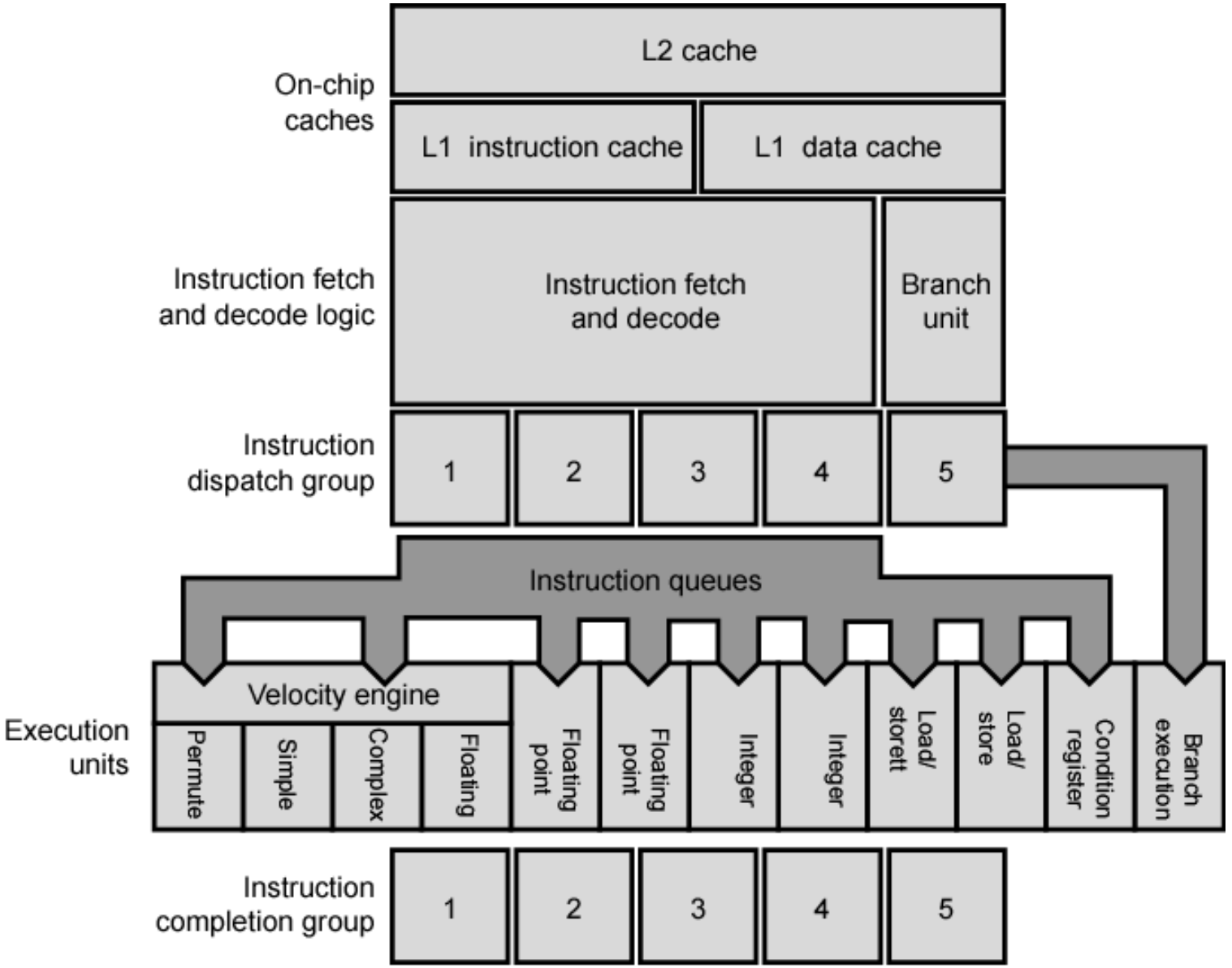
Pentium 4 Design Reasoning

- Decodes instructions into RISC like micro-ops before L1 cache
- Micro-ops fixed length
  - Superscalar pipelining and scheduling
- Pentium instructions long & complex
- Performance improved by separating decoding from scheduling & pipelining
  - (More later – ch14)
- Data cache is write back
  - Can be configured to write through
- L1 cache controlled by 2 bits in register
  - CD = cache disable
  - NW = not write through
  - 2 instructions to invalidate (flush) cache and write back then invalidate
- L2 and L3 8-way set-associative
  - Line size 128 bytes

PowerPC Cache Organization

- 601 – single 32kb 8 way set associative
- 603 – 16kb (2 x 8kb) two way set associative
- 604 – 32kb
- 620 – 64kb
- G3 & G4
  - 64kb L1 cache
    - 8 way set associative
  - 256k, 512k or 1M L2 cache
    - two way set associative
- G5
  - 32kB instruction cache
  - 64kB data cache

PowerPC G5 Block Diagram



Internet Sources

- Manufacturer sites
  - Intel
  - IBM/Motorola
- Search on cache