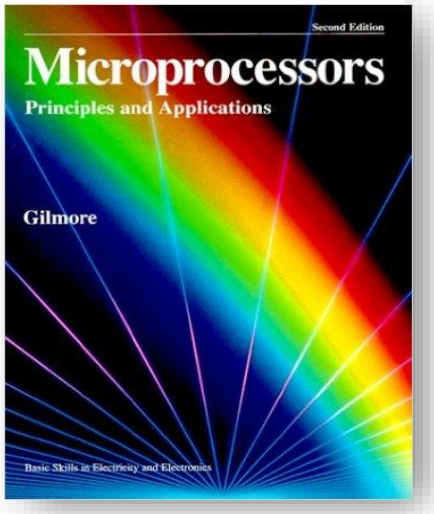


Nội dung

- Tổng quan
- Cấu trúc các bộ vi xử lí
- Bộ vi xử lí Intel 8088/8086
- Lập trình Hợp ngữ với 8088/8086
- Ghép nối 8088/8086 với bộ nhớ và thiết bị ngoại vi
- Tổ chức vào ra dữ liệu

Tài liệu tham khảo

- [1] Văn Thế Minh, *Kỹ thuật vi xử lý*, NXB Giáo dục 1997.
- [2] Đỗ Xuân Thụ, Hồ Khánh Lâm, *Kỹ thuật Vi xử lý và máy vi tính*, NXB Giáo dục 2000.
- [3] Quách Tuấn Ngọc, *Ngôn ngữ lập trình Assembly và máy vi tính IBM-PC*, 2 tập, NXB Giáo dục, 1995.
- [4] Charles M.Gilmore McGraw, *Microprocessors Principles and Application*, 2nd Edition, Hill International Edition 1995.
- [5] William Stallings, *Computer Organization and Architecture*, Fifth edition, Prentice Hall, 2000.

Tổng quan

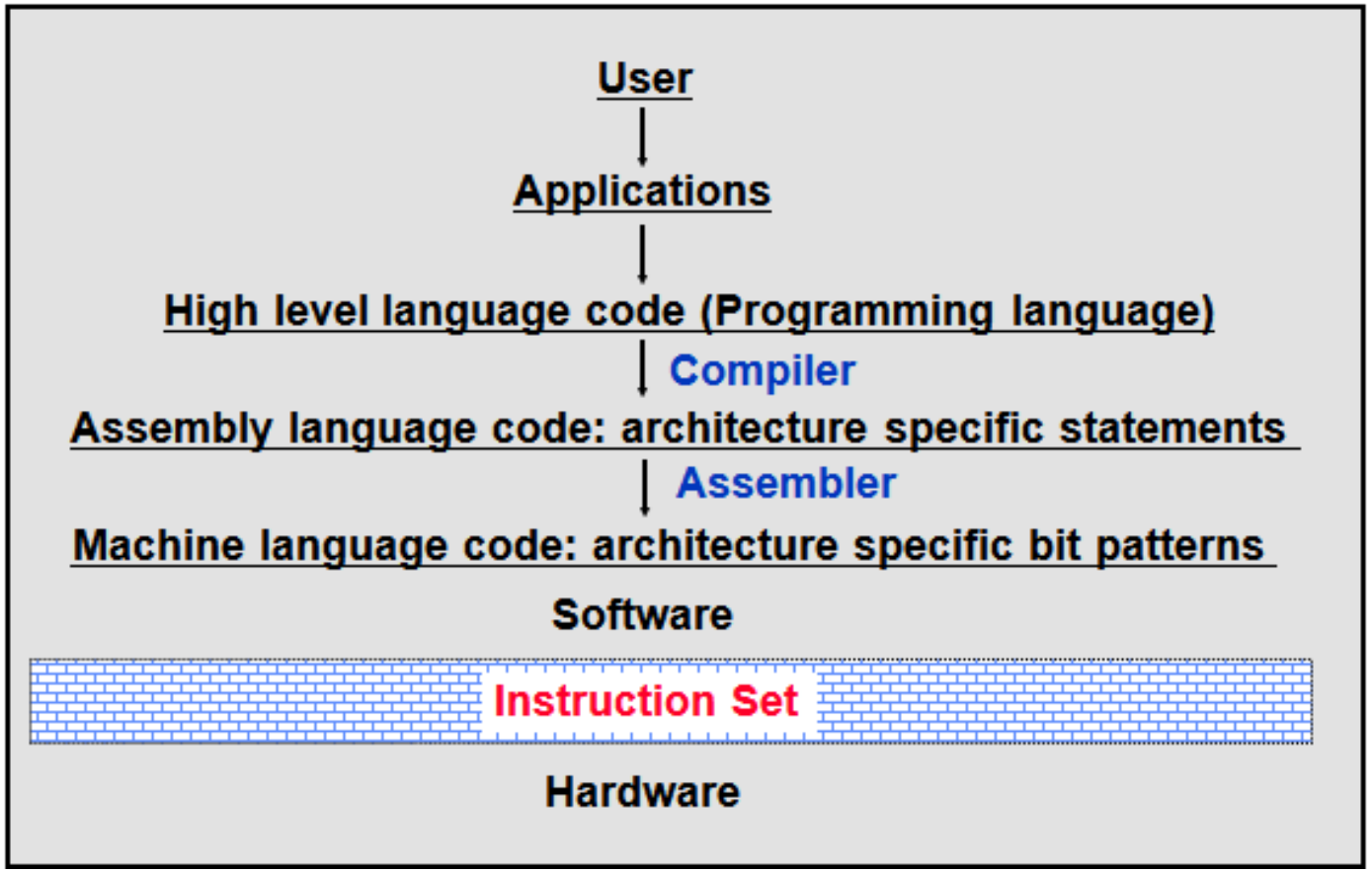
- Tổng quan
- Lịch sử phát triển và phân loại
- Cấu trúc và hoạt động của hệ vi xử lí
- Những đặc điểm cấu trúc

Tổng quan

- Tổng quan
  - Máy tính (computer): thiết bị điện tử có khả năng thao tác (lưu trữ, xử lý) trên các thông tin (dữ liệu).
  - Các thao tác: thực hiện theo một chương trình - dãy các câu lệnh

Tổng quan

- Giao tiếp người - máy

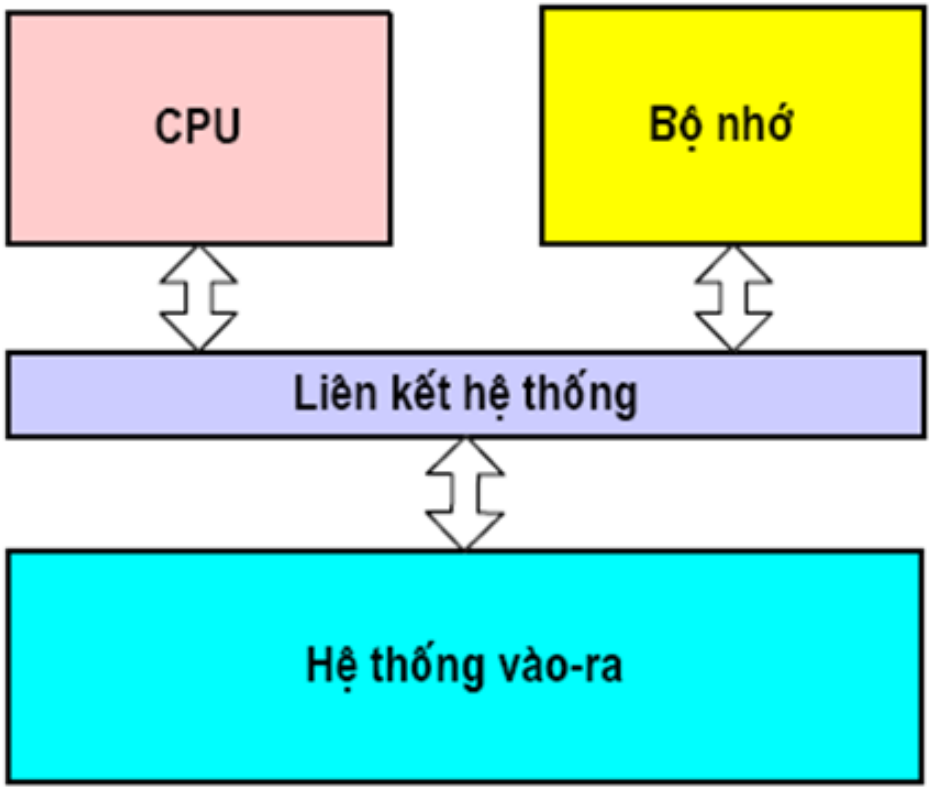


Tổng quan

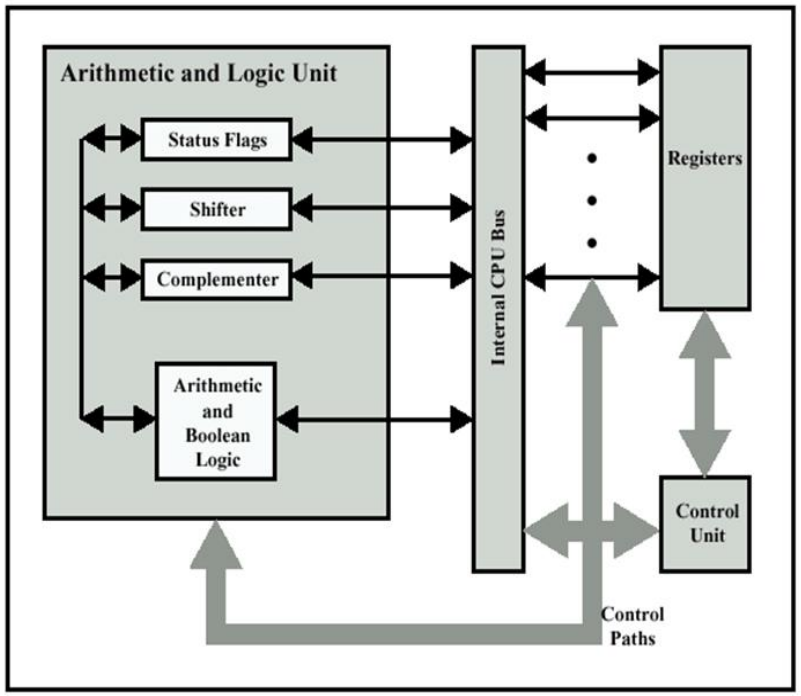
- Tổ chức máy tính: cấu trúc phần cứng của máy tính
- Bao gồm:
  - Bộ xử lý trung tâm (Central Processing Unit): điều khiển hoạt động của máy tính theo đúng lệnh, thứ tự lệnh
  - Hệ thống nhớ (Memory): lưu trữ dữ liệu và chương trình
  - Hệ thống vào ra (Input/Output System): trao đổi thông tin giữa máy tính với môi trường ngoài
  - Cấu trúc kết nối (Connection Structure): liên kết các thành phần trong hệ thống

Tổng quan

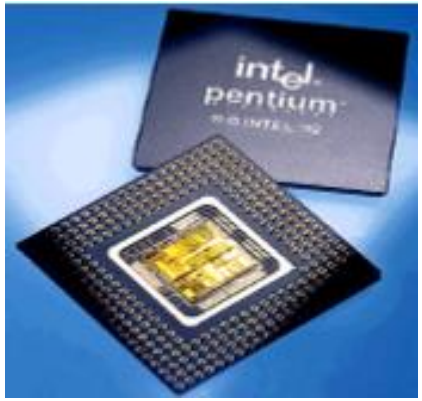
- Kiến trúc máy tính



Tổng quan



Central Processing Unit - CPU



Microprocessor - µP

Lịch sử phát triển

Máy tính ENIAC:

- Dự án của bộ quốc phòng Mỹ
- Bắt đầu năm 1943, kết thúc năm 1946
- Đặc điểm

  Nặng 30 tấn,

  18.000 đèn điện tử

  1500 rơle

  Công suất tiêu thụ 140KW

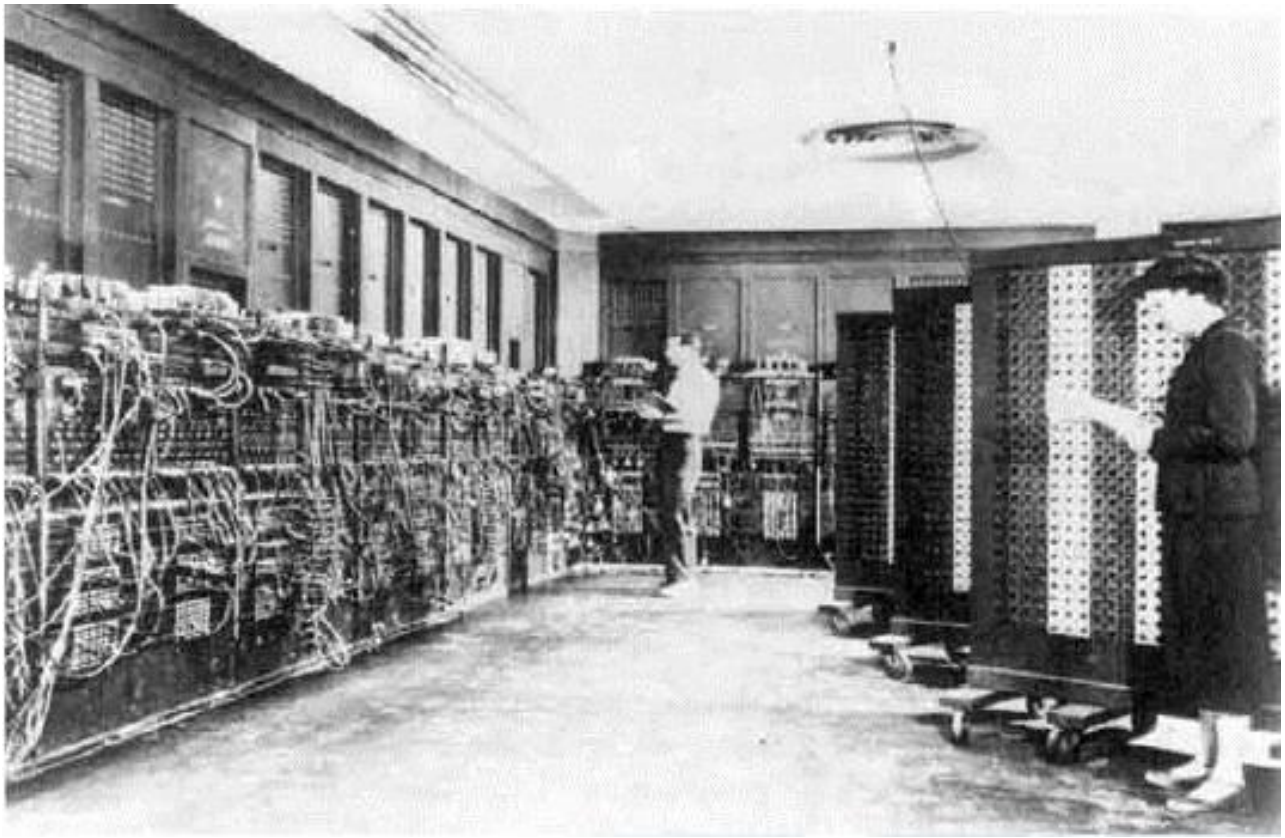
  Tốc độ: 5000 phép cộng mỗi giây

  Bộ nhớ chỉ lưu trữ dữ liệu

  Lập trình bằng cách thiết lập các chuyển mạch và các cáp nối

Lịch sử phát triển

Máy tính ENIAC:



Lịch sử phát triển

- Thế hệ 1: Máy tính dùng đèn điện tử chân không (1946 - 1955)
  - Sử dụng công nghệ đèn điện tử chân không➡độ tin cậy thấp, tổn hao năng lượng. Tốc độ tính toán từ vài nghìn đến vài trăm nghìn phép tính/giây
  - Phần mềm: ngôn ngữ máy
  - Ứng dụng: khoa học, kỹ thuật

Lịch sử phát triển

- Thế hệ 2: Máy tính dùng transistor (1956 - 1965)
  - Sử dụng linh kiện bán dẫn (transistor). Bộ nhớ được làm bằng xuyến từ
  - Phần mềm: sử dụng một số ngôn ngữ lập trình bậc cao: Fortran, Algol, Cobol, ... Xuất hiện các hệ điều hành tuần tự
  - Ứng dụng: các bài toán kinh tế

Lịch sử phát triển

- Thế hệ 3: Máy tính dùng mạch tích hợp (1966 - 1980)
  - Sử dụng mạch tích hợp (IC), các thiết bị ngoại vi được cải tiến, đĩa từ được sử dụng rộng rãi➡ Tốc độ tính toán đạt vài triệu phép toán trên giây
  - Phần mềm: xuất hiện nhiều hệ điều hành khác nhau, đa dạng➡chất lượng cao, cho phép khai thác máy tính theo nhiều chế độ
  - Ứng dụng: nhiều lĩnh vực

Lịch sử phát triển

- Thế hệ 4: Máy tính dùng mạch tích hợp cỡ lớn VLSI, ULSI (1981- nay)
  - Sử dụng mạch tích hợp cỡ lớn (VLSI - Very large scale integration), các cấu trúc đa xử lý
  - Các hệ thống bộ nhớ bán dẫn, bộ nhớ ảo, bộ nhớ cache được sử dụng rộng rãi
  - Các kỹ thuật cải tiến tốc độ vi xử lý: vô hướng, ống dẫn, xử lý song song…➡ Tốc độ đạt tới hàng chục triệu phép tính/giây

Sự phát triển của các thế hệ VXL Intel

- 4004:
  - Bộ vi xử lý đầu tiên
  - 4 bít
- 8080
  - Bộ xử lý đa năng đầu tiên
  - Bus dữ liệu ngoài: 8 bit
- 8086
  - 5Mhz, tích hợp 29,000 transistor
  - Bus dữ liệu ngoài: 16 bit

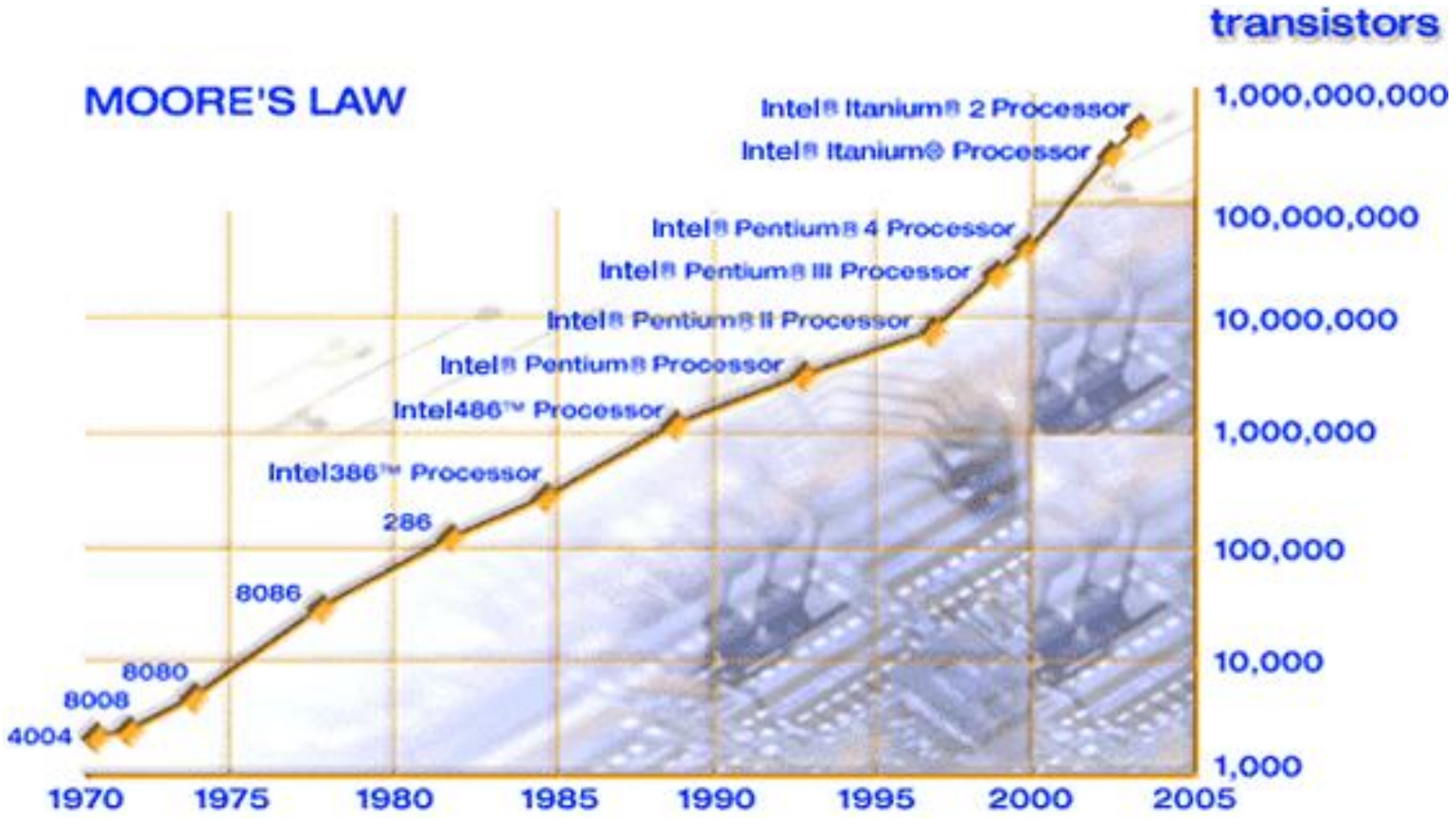
Sự phát triển của các thế hệ VXL Intel

- 80286: bộ nhớ 16Mbyte
- 80386: 32 bit, hỗ trợ đa nhiệm
- 80486:
  - Hỗ trợ pipe line
  - Tích hợp bộ đồng xử lý toán học
- Pentium
  - Siêu vô hướng
  - 64 bit
  - Đa lệnh thực hiện đồng thời
- Pentium Pro
  - Dự đoán rẽ nhánh
  - Kĩ thuật đa luồng
  - Lập lịch động

Sự phát triển của các thế hệ VXL Intel

- Pentium II: xử lý đồ họa, video, audio
- Pentium III: thêm các lệnh xử lý dấu chấm động
- Pentium IV: hỗ trợ multimedia
- Dual core: tích hợp 2 VXL/chip
- Core 2 dual: kiến trúc 64 bit
- Core 2 quard: 4 VXL/chip
- Core iX
- …

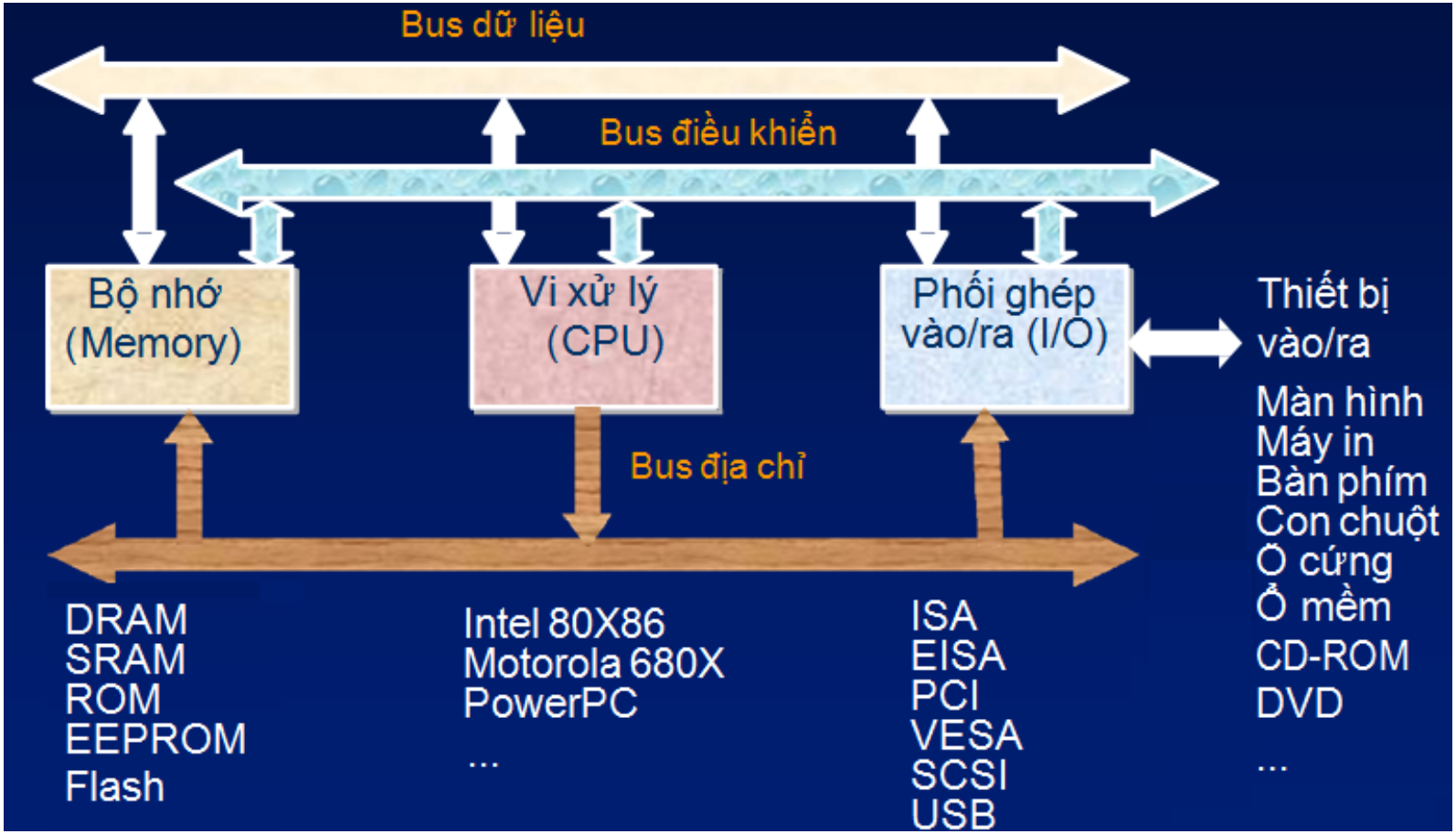
Sự phát triển của các thế hệ VXL Intel



Phân loại vi xử lí

- Phân loại chức năng
  - Vi xử lý đa năng (General Purpose Microprocessor)
  - DSP (Digital Signal Processor)
  - Vi điều khiển (Microcontroller)
  - ASIP (Application Specific Integrated Processor)
  - …
- Phân loại theo tập lệnh
  - CISC (Complex Instruction Set Computer)
  - RISC (Reduced Instruction Set Computer)
  - MIPS (Microprocessor without Interlocked Pipeline Stages)

Hệ vi xử lí


Bus dữ liệu
Bus điều khiển
Bộ nhớ (Memory)
Vi xử lý (CPU)
Phối ghép vào/ra (I/O)
Thiết bị vào/ra
Màn hình
Máy in
Bàn phím
Con chuột
Ổ cứng
Ổ mềm
CD-ROM
DVD
...
Bus địa chỉ
DRAM
SRAM
ROM
EEPROM
Flash
Intel 80X86
Motorola 680X
PowerPC
...
ISA
EISA
PCI
VESA
SCSI
USB

Khả năng xử lí dữ liệu

- Độ dài từ nhớ
- Khả năng đánh địa chỉ
- Tốc độ thực hiện lệnh

$$MIPS = \frac{f * N}{M + T}$$

  - f: tần số làm việc của bộ VXL
  - N: số lượng các đơn vị xử lý (ALU) không phụ thuộc vào nhau bên trong bộ vi xử lý
  - M: số lượng vi lệnh (micro-instruction) trung bình của 1 lệnh
  - T: hệ số thời gian truy nhập bộ nhớ (chu trình chờ đợi trong khi truy nhập bộ nhớ

Hiệu năng hệ thống

$$\text{Perf} = \frac{1}{\text{Exe time}}$$

- Perf (perfomance): hiệu năng hệ thống
- Exe time (execute): thời gian thực hiện

$$\text{CPU time} = \text{CPU clock cycles} \times \text{Cycle time} = \frac{\text{CPU clock cycles}}{\text{Clock rate}}$$

- CPU time: thời gian thực hiện cho 1 chương trình
- CPU clock cycle: số chu kì cho 1 chương trình
- Cycle time: thời gian cho 1 chu kì
- Clock rate: tốc độ xử lí của CPU (tần số hoạt động của CPU)

Hiệu năng hệ thống

VD1: Một chương trình chạy trên máy tính A (tốc độ 400MHz) trong 10 giây. Cũng chương trình ấy khi chạy trên máy tính "B" hết 6 giây. Tính tốc độ máy "B" biết số chu kì dùng cho chương trình khi chạy trên máy này gấp 1.2 lần khi chạy trên máy "A".

Với máy "A":

$$\text{CPU time (A)} = \frac{\text{CPU clock cycles}}{\text{Clock rate (A)}} \qquad 10\,\text{s} = \frac{\text{CPU clock cycles}}{400 \times 10^6\ \text{cycles/s}}$$

$$\text{CPU clock cycles} = 10\,\text{s} \times 400 \times 10^6\ \text{cycles/s} = 4000 \times 10^6\ \text{cycles}$$

Tương tự với máy "B":

$$6\,\text{s} = \frac{1.2 \times \text{CPU clock cycles}}{\text{clock rate (B)}} = \frac{1.2 \times 4000 \times 10^6\ \text{cycles}}{\text{clock rate (B)}}$$

$$\text{Clock rate (B)} = \frac{1.2 \times 4000 \times 10^6\ \text{cycles}}{6\,\text{s}} = 800 \times 10^6\ \text{cycles/s}$$

Hiệu năng hệ thống

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{Cycle time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

- CPU time: thời gian thực hiện cho 1 chương trình
- Instruction count: số lệnh cần thực thi cho chương trình
- CPI: số chu kì cho 1 lệnh
- Cycle time: thời gian cho 1 chu kì
- Clock rate: tốc độ xử lí của CPU (tần số hoạt động của CPU)

Hiệu năng hệ thống

VD2: Hai máy "A" và "B" có tập lệnh tương tự, thời gian cho 1 chu kì của "A" là 1ns và chỉ số CPI của nó là 2.0. Máy "B" thời gian cho 1 chu kì là 2ns và chỉ số CPI của nó là 1,2. So sánh tốc độ thực hiện của "A" và "B"?.

Giả sử số lệnh cần thực hiện cho 1 chương trình của "A" và "B" là i. Với máy "A":

$$\text{CPU time (A)} = \text{Instruction count} \times \text{CPI} \times \text{Cycles time} = \text{i} \times 2 \times 1ns$$

Tương tự với máy "B":

$$\text{CPU time (B)} = \text{Instruction count} \times \text{CPI} \times \text{Cycles time} = \text{i} \times 1.2 \times 2ns$$

Tương quan tốc độ giữa "A" và "B":

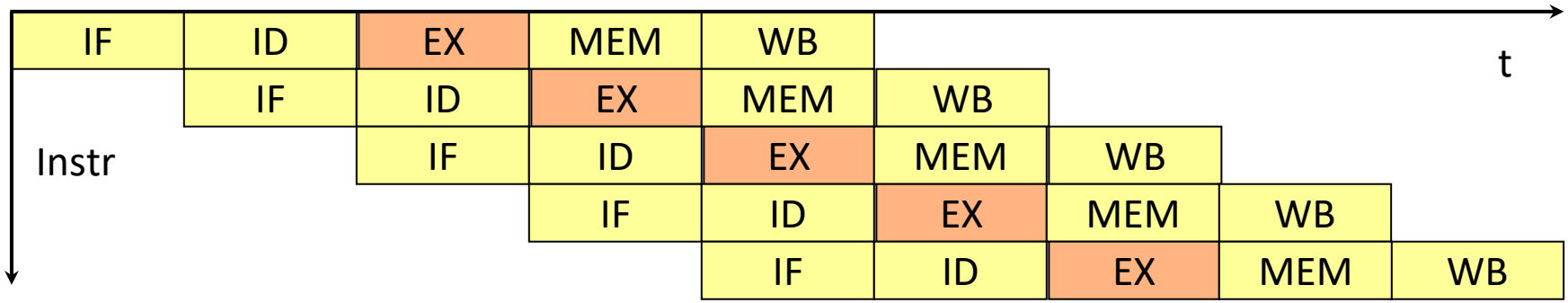
$$\frac{\text{Perf (A)}}{\text{Perf (B)}} = \frac{\text{CPU time (B)}}{\text{CPU time (A)}} = \frac{\text{i} \times 1.2 \times 2\text{ns}}{\text{i} \times 2 \times 1\text{ns}} = 1.2$$

Các giải pháp tăng hiệu năng hệ thống

- Giải pháp kĩ thuật
  - Xử lí song song
  - Sử dụng các bộ đồng xử lí
  - Kĩ thuật bộ nhớ Cache
- Giải pháp công nghệ
  - Pipeline
  - Superscalar

Các giải pháp tăng hiệu năng hệ thống

- Pipeline

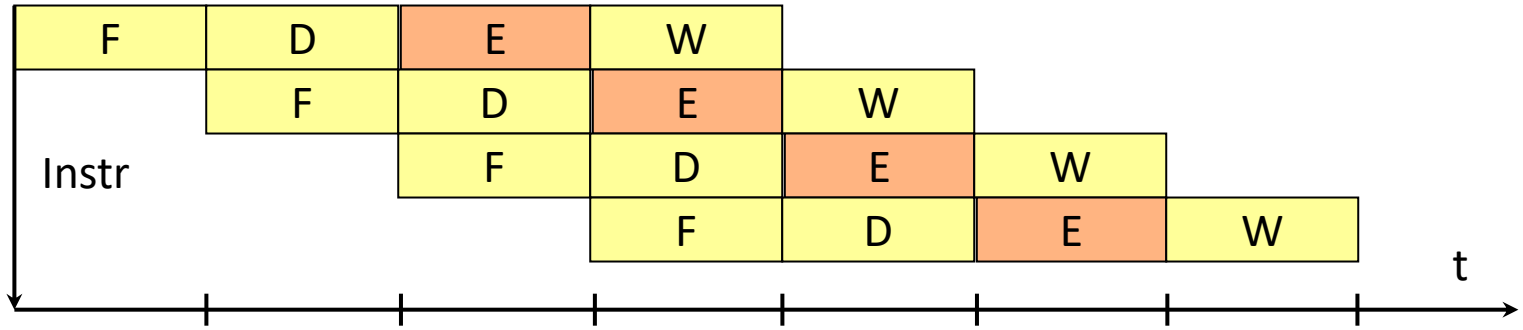


Giả sử 1cycle = 0.5ns

- Tuần tự: 5 instr = 5*5*0.5 = 12.5 ns
- Pipeline: 5 instr = 9*0.5 = 4.5 ns

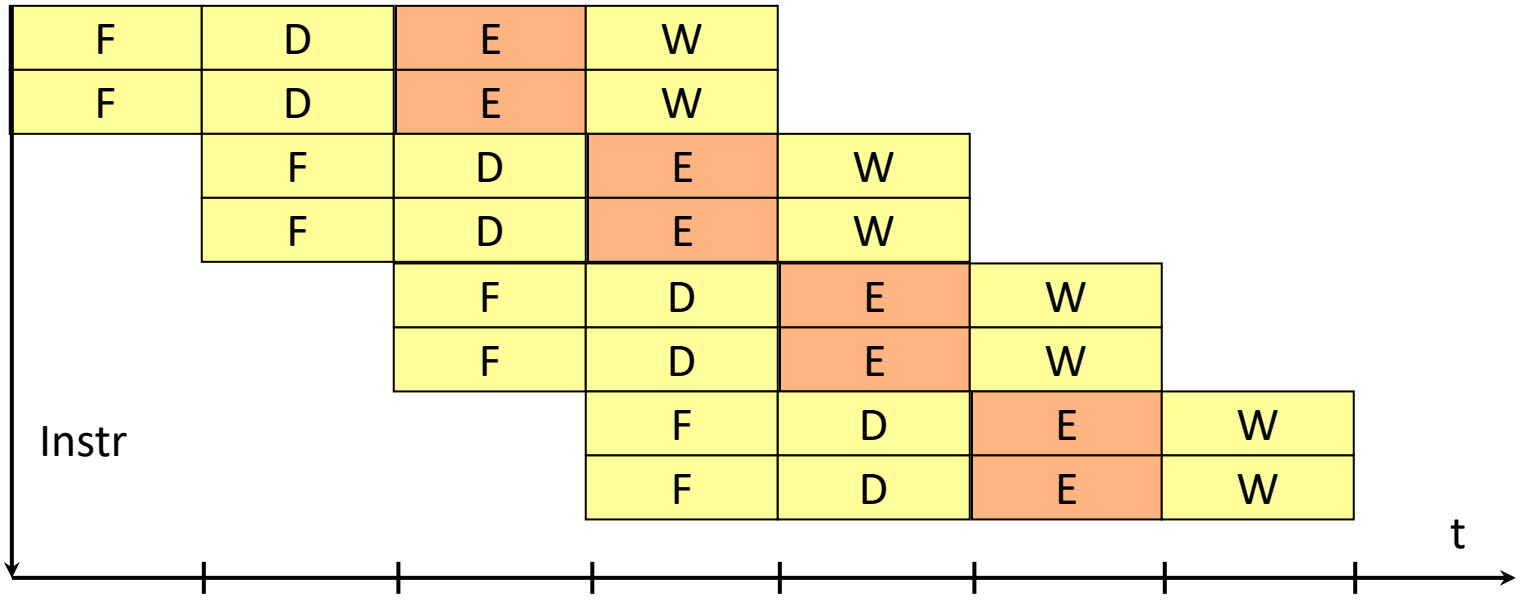
Các giải pháp tăng hiệu năng hệ thống

- Scalar
  - Mỗi 1 chu kì chỉ đưa ra (issue) 1 lệnh, các lệnh được đưa ra theo thứ tự - kiến trúc RISC
  - CPU cơ bản là một bộ xử lý vô hướng bao gồm nhiều đơn vị chức năng
  - Tích hợp đơn vị xử lí dấu phẩy động (FPU), bộ đồng xử lí (Co-processor)



Các giải pháp tăng hiệu năng hệ thống

- Superscalar
  - Mỗi 1 chu kì có thể đưa ra nhiều lệnh
  - Các lệnh có thể được xử lí đồng thời tại các đơn vị thực hiện khác nhau - song song mức lệnh



Hệ đếm và cơ số đếm

- Hệ đếm
  - Hệ đếm cơ số bất kì

$$N = a_{n-1}\ldots a_0.b_1b_2\ldots b_m = a_0.s^0 + a_1.s^1 + \ldots + a_{n-1}.s^{n-1} + b_1.s^{-1} + b_2.s^{-2} + \ldots + b_m.s^m$$

  - Trong đó
    - N: số nguyên bao gồm n+m chữ số
    - s: cơ số của hệ đếm (2, 8, 10, 16…)
    - a_i, b_j (0 ÷ s-1): giá trị của phần tử thứ i, j
    - i (0 ÷ n-1), j (1 ÷ m): trọng số

Hệ đếm và cơ số đếm

- Hệ đếm
  - Hệ đếm cơ số 2 (binary)
    - s = 2
    - a_i, b_j (0, 1)

    Ví dụ:

    $1101001.1011_{(2)} = 2^6 + 2^5 + 2^3 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4}$

    $= 64 + 32 + 8 + 1 + 0.5 + 0.125 + 0.0625$

    $= 105.6875_{(10)}$
  - Hệ đếm cơ số 10 (decimen)
    - s = 10
    - a_i, b_j (0 ÷ 9)
  - Hệ đếm cơ số 16 (hexa)
    - s = 16
    - a_i (0÷ 9, A ÷ F)

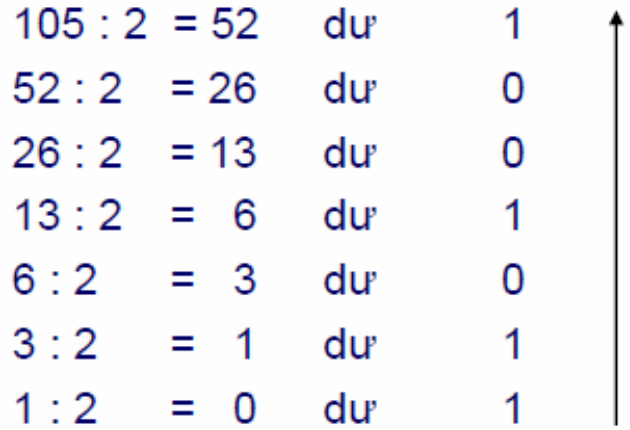
Hệ đếm và cơ số đếm

- Chuyển đổi giữa các hệ cơ số
  - Hệ cơ số 2 sang hệ cơ số 10: theo công thức tổng quát
  - Hệ cơ số 10 sang hệ 2:
    - Phần nguyên: chia liên tiếp cho 2 tới khi gặp thương số là 0, lấy tổ hợp các số dư theo chiều ngược lại
    - Phần thập phân: nhân liên tiếp với 2, lấy tổ hợp các phần nguyên ở mỗi lần nhân theo chiều thuận

  Ví dụ: đổi số 105.6875 sang hệ 2

Hệ đếm và cơ số đếm

- Chuyển đổi giữa các hệ cơ số
  - Hệ cơ số 10 sang hệ 2:
    - Phần nguyên:



    - Phần thập phân:

0.6875	x 2	=	1.3750	phần nguyên = 1
0.375	x 2	=	0.750	phần nguyên = 0
0.75	x 2	=	1.50	phần nguyên = 1
0.5	x 2	=	1.0	phần nguyên = 1

Hệ đếm và cơ số đếm

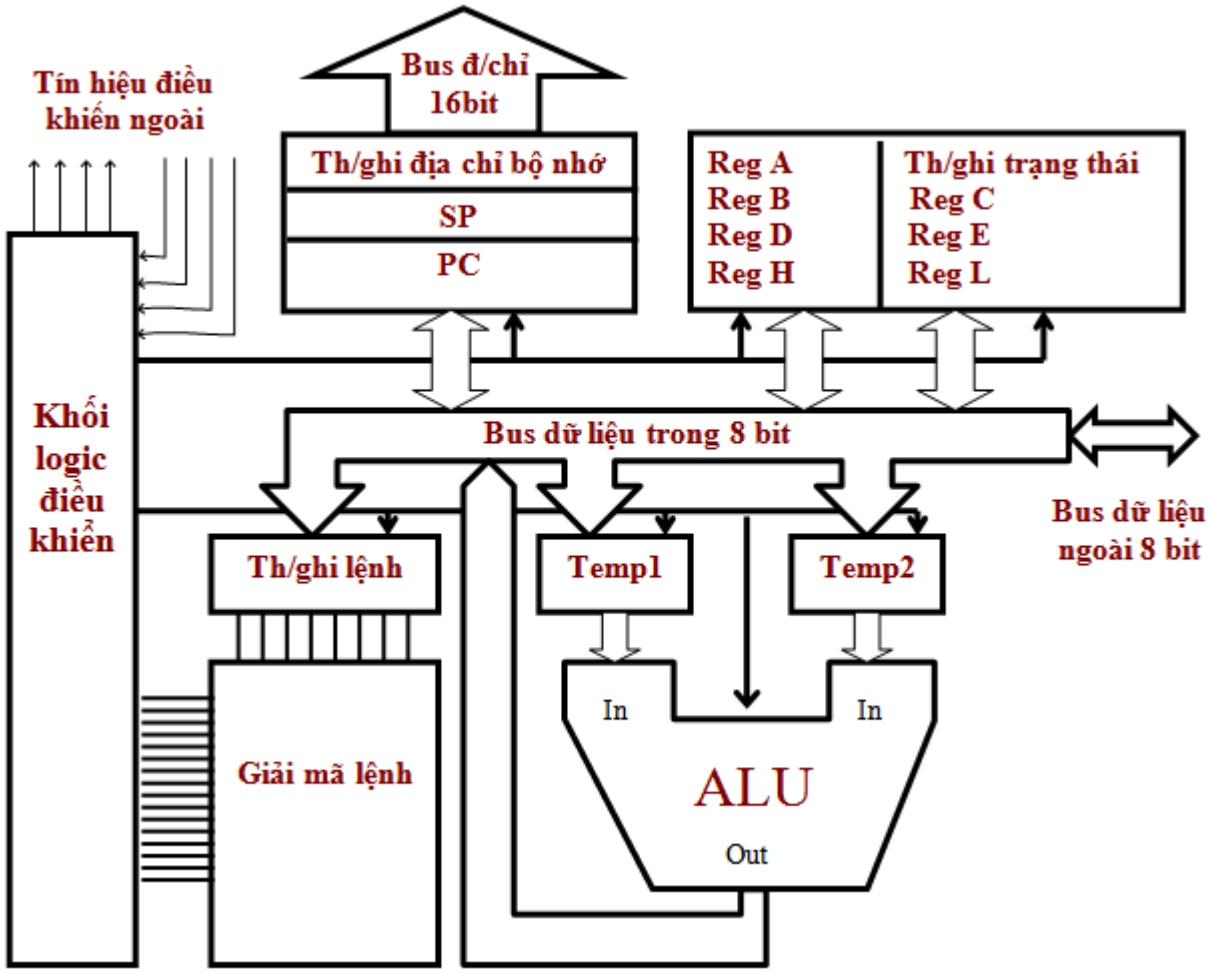
- Chuyển đổi giữa các hệ cơ số
  - Quan hệ giữa các hệ cơ số 2, 10 và 16

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

Cấu trúc các bộ vi xử lí

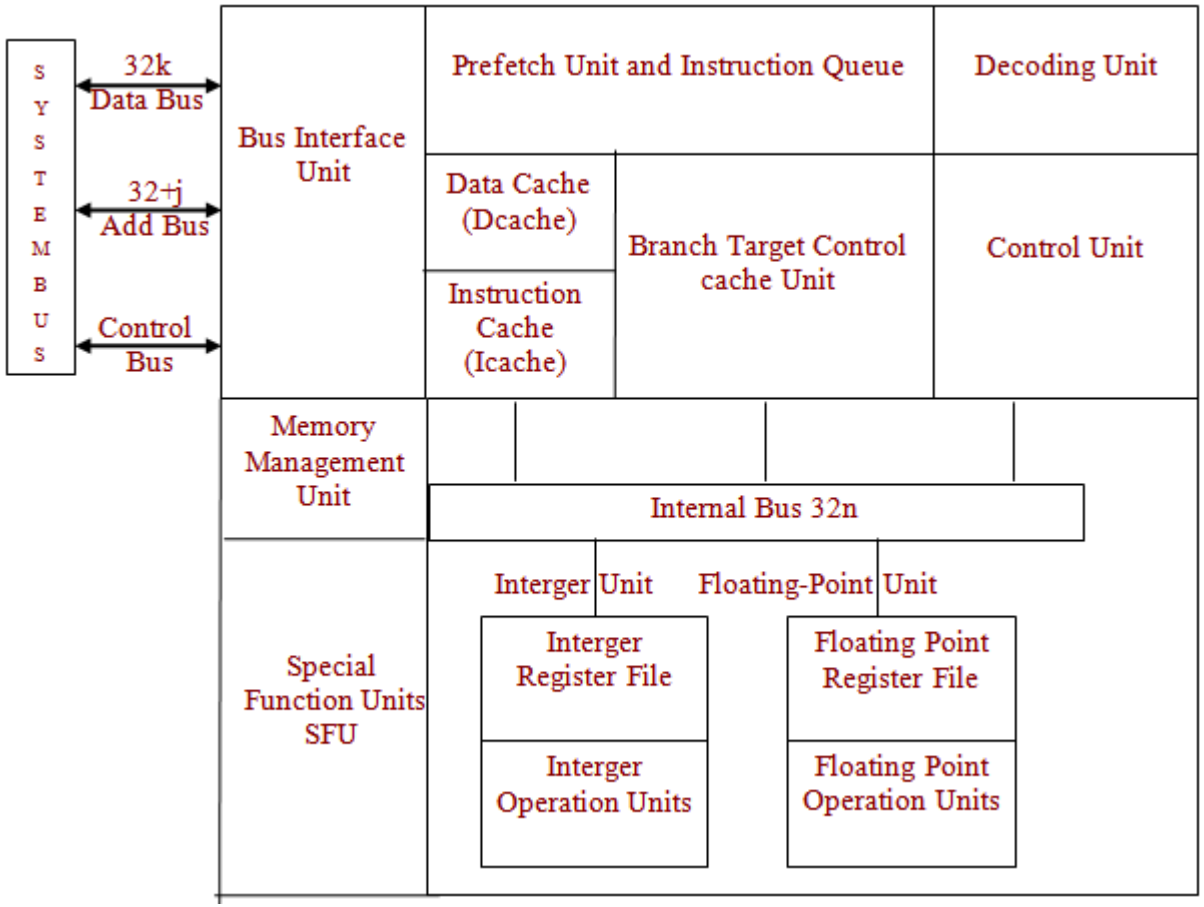
- Bộ xử lí cấp thấp
- Bộ xử lí cấp cao

Cấu trúc các bộ vi xử lí



Cấu trúc điển hình của các bộ xử lí cấp thấp

Cấu trúc các bộ vi xử lí

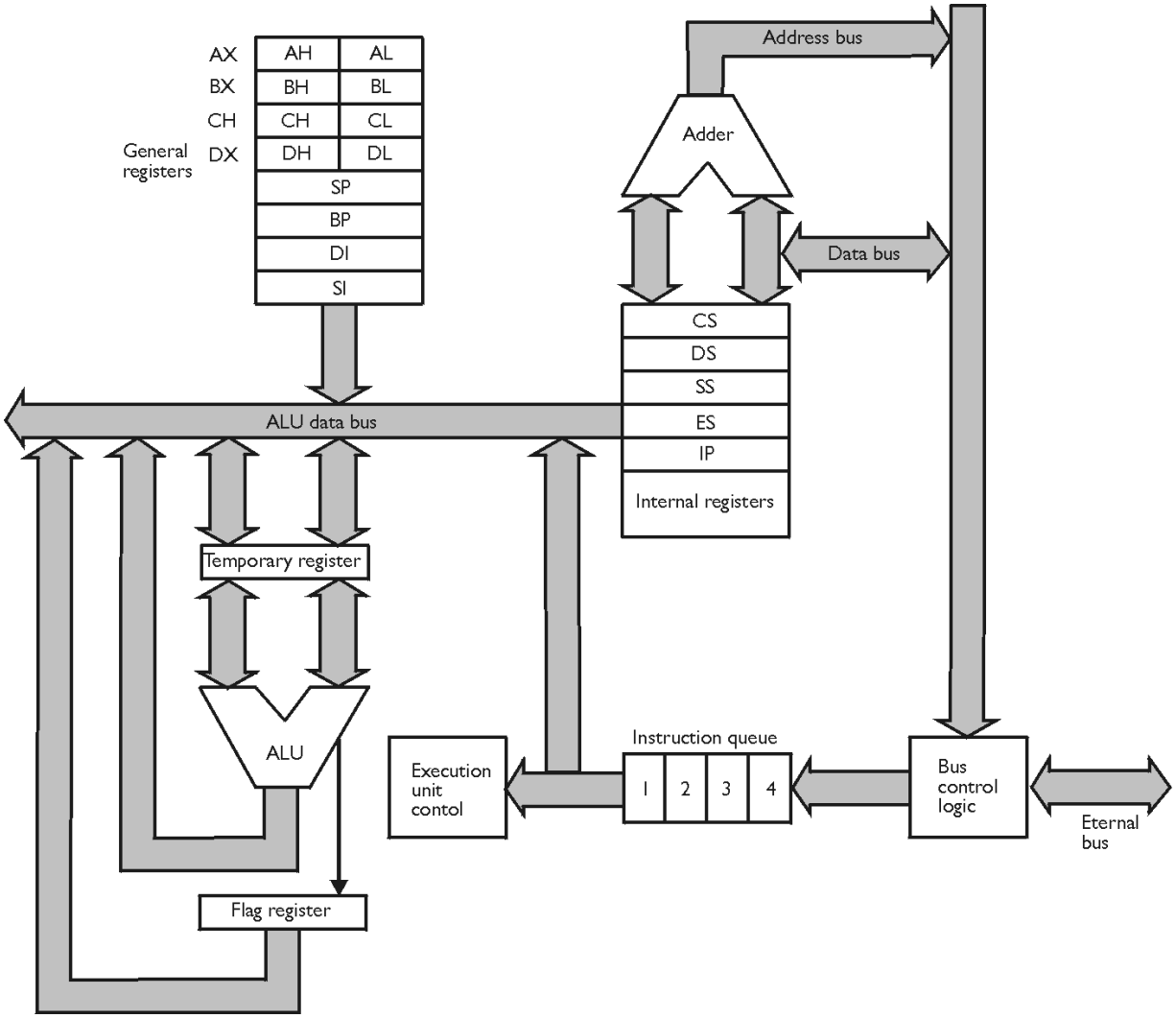


Cấu trúc điển hình của các bộ xử lí cấp cao

Bộ vi xử lí Intel 8088/8086

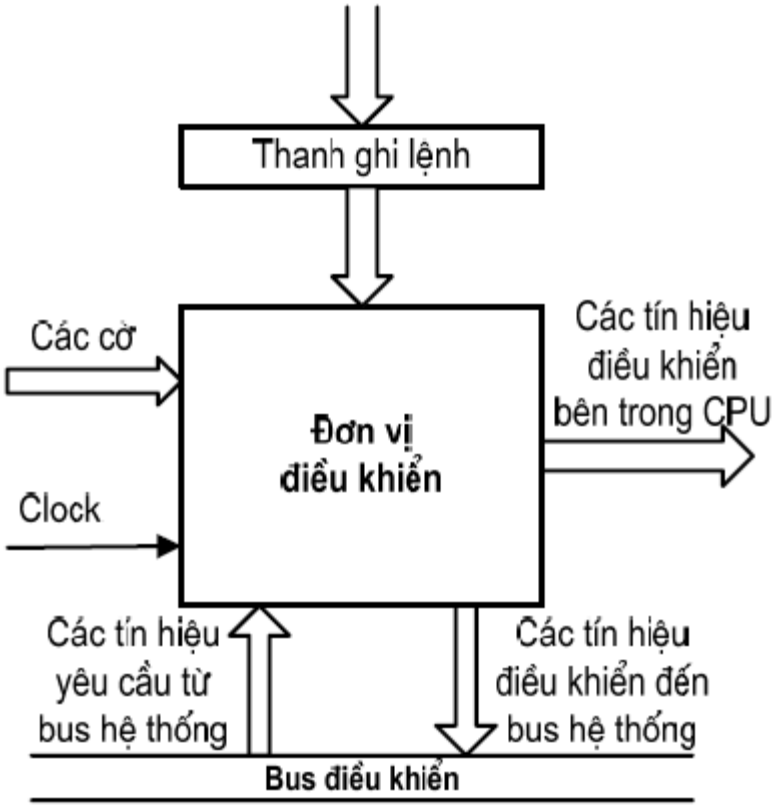
- Cấu trúc trong
- Các chân tín hiệu
- Bản đồ bộ nhớ của máy tính IBM-PC
- Các chế độ địa chỉ của 8086
- Cách mã hoá lệnh của 8086
- Tập lệnh của 8086

Bộ vi xử lí Intel 8088/8086



Cấu trúc trong của 8086

Đơn vị điều khiển

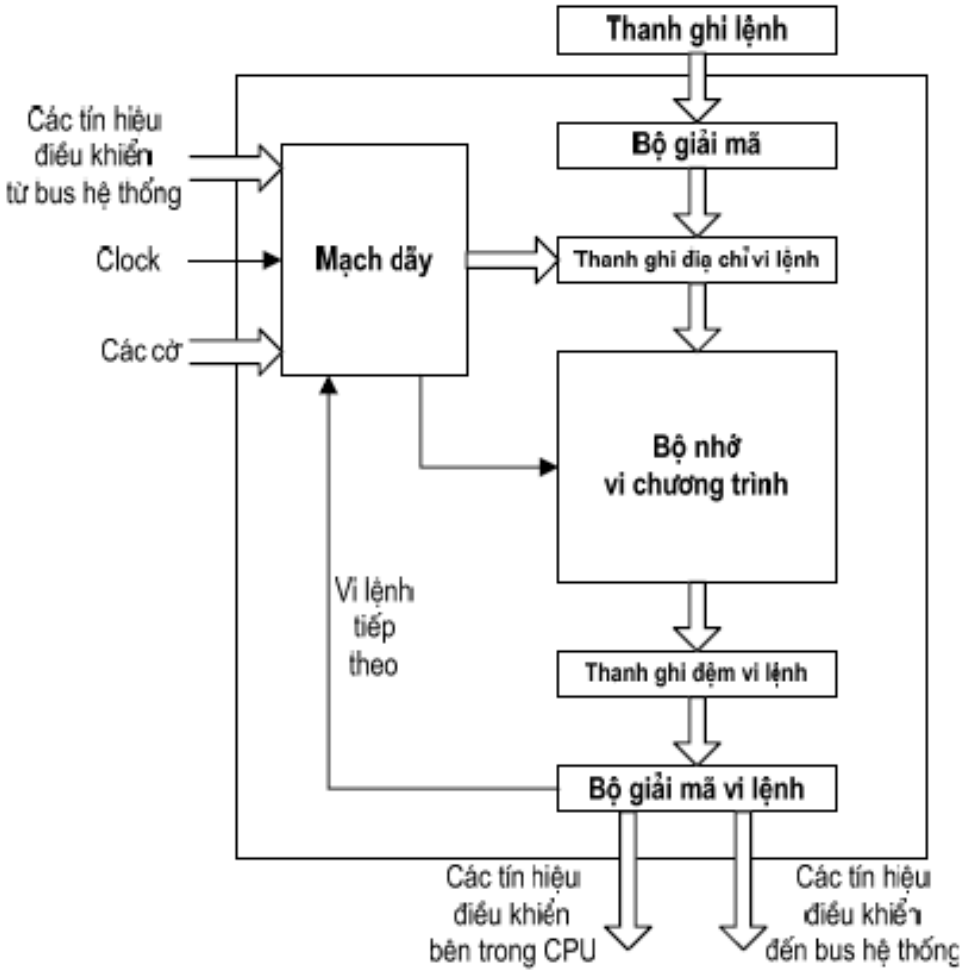


Các tín hiệu vào/ra CU

- Điều khiển việc nhận lệnh từ bộ nhớ ➡ thanh ghi lệnh (IP)
- Tăng nội dung của thanh ghi PC ➡ lệnh kế tiếp
- Giải mã lệnh ➡ xác định thao tác mà lệnh yêu cầu
- Phát tín hiệu điều khiển thực hiện lệnh
- Nhận các tín hiệu yêu cầu từ bus hệ thống ➡ đáp ứng

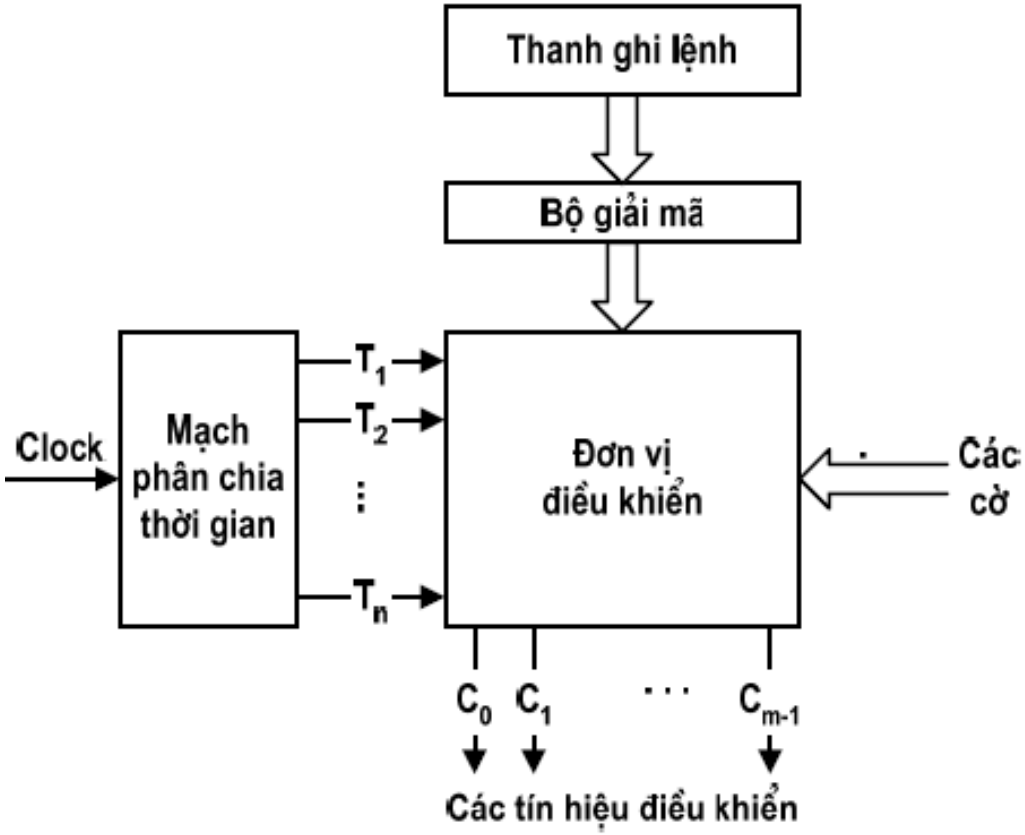
Đơn vị điều khiển

Đơn vị điều khiển bằng phần mềm:



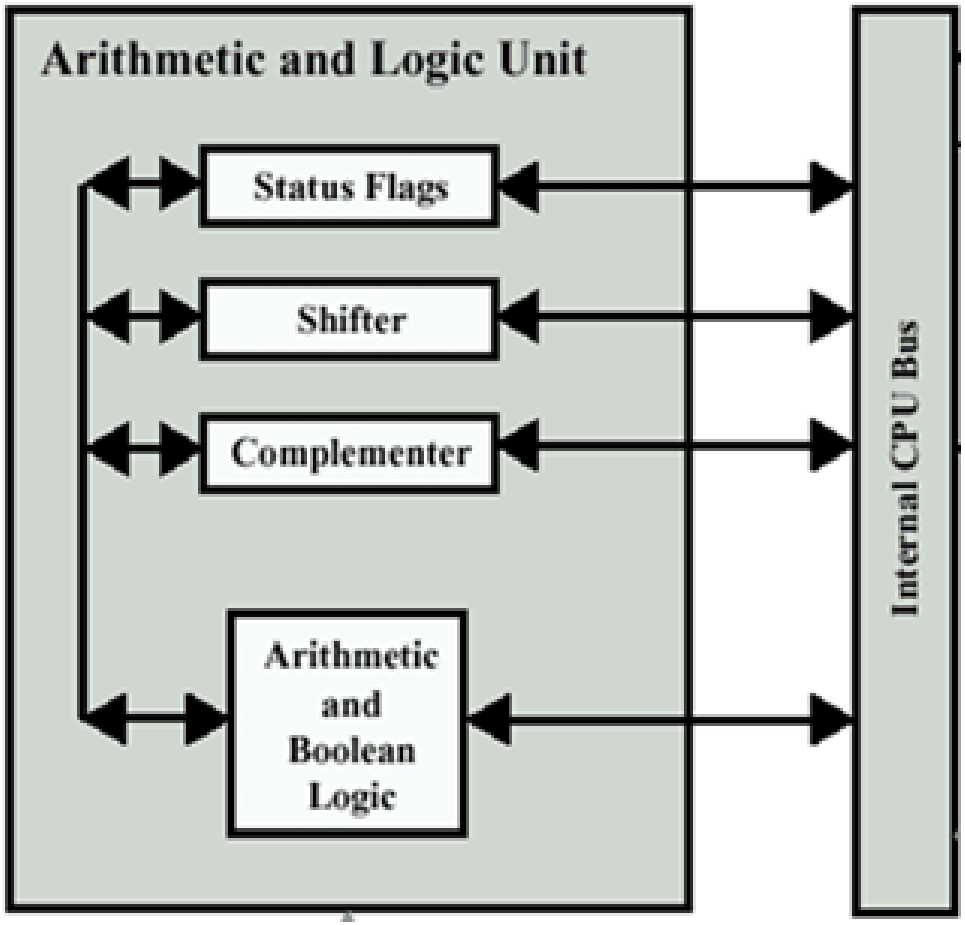
Đơn vị điều khiển

Đơn vị điều khiển bằng phần cứng:



Đơn vị số học và logic

Thực hiện các phép toán số học và logic



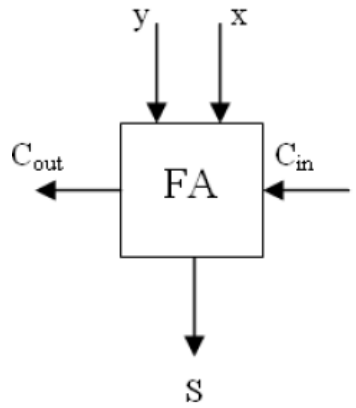
Đơn vị số học và logic

- Các phép toán số học: cộng, trừ, nhân, chia
  - Phép cộng (Addition)

x	y	C_{in}	S	C_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$S = x \oplus y \oplus C_{in}$$

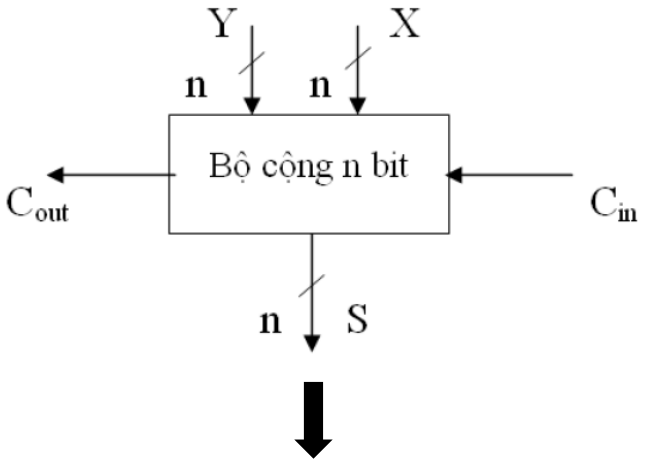
$$C_{out} = x.y + x.C_{in} + y.C_{in}$$

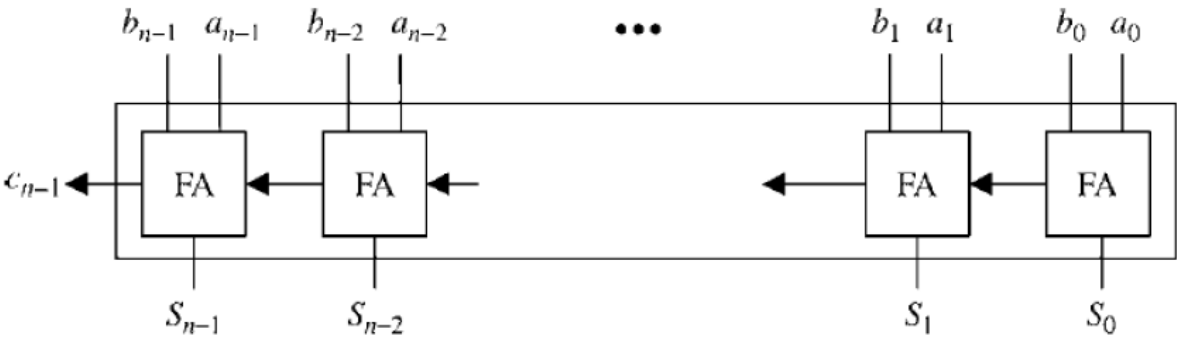


Bộ cộng

Đơn vị số học và logic

- Các phép toán số học: cộng, trừ, nhân, chia
  - Phép cộng (Addition)

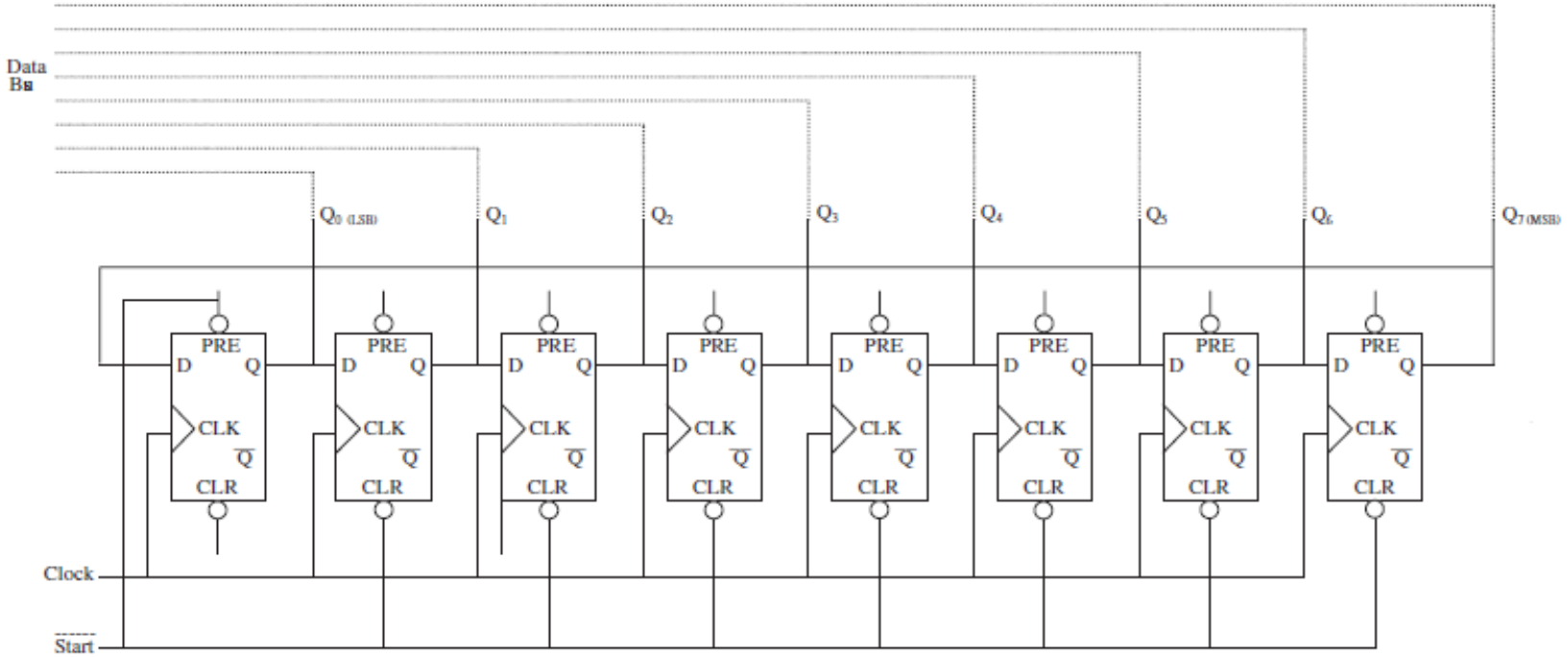




Mạch cộng

Thanh ghi

- Thanh ghi (Register): lưu trữ tạm thời lệnh và dữ liệu trong quá trình xử lí
- Được hình thành từ các phần tử nhớ cơ bản FF
- Số FF quy định độ rộng thanh ghi, VXL



Thanh ghi 8 bit

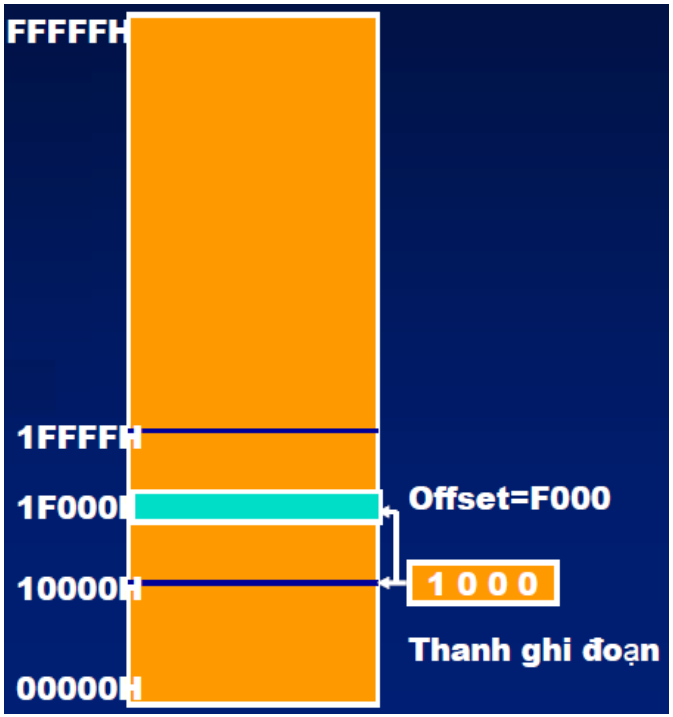
Thanh ghi

- Các thanh ghi đa năng
  - AX (Accumulator): chứa kết quả của các phép tính, kết quả 8 bit được chứa trong AL
  - BX (Base): chứa địa chỉ cơ sở
  - CX (Count): chứa số lần lặp trong các lệnh lặp (Loop). CL được dùng để chứa số lần dịch hoặc quay trong các lệnh dịch và quay thanh ghi
  - DX (Data): cùng AX chứa dữ liệu trong các phép tính nhân chia số 16 bit. DX còn được dùng để chứa địa chỉ cổng trong các lệnh vào ra dữ liệu trực tiếp (IN/OUT)

Thanh ghi

- Các thanh ghi đoạn
  - Bộ nhớ được chia thành các phần: đoạn (segment), trỏ bởi các thanh ghi đoạn
  - Địa chỉ:
    - Địa chỉ đoạn
    - Độ lệch (offset)

➡ Địa chỉ vật lí = Thanh ghi đoạn * 16 + Offset



Thanh ghi

- Các thanh ghi đoạn

  Ví dụ 1: Địa chỉ vật lý 12345H có thể được tạo ra từ các giá trị:

Thanh ghi đoạn	Offset
1000H	2345H
1200H	0345H
1004H	2305H
0300H	F345H

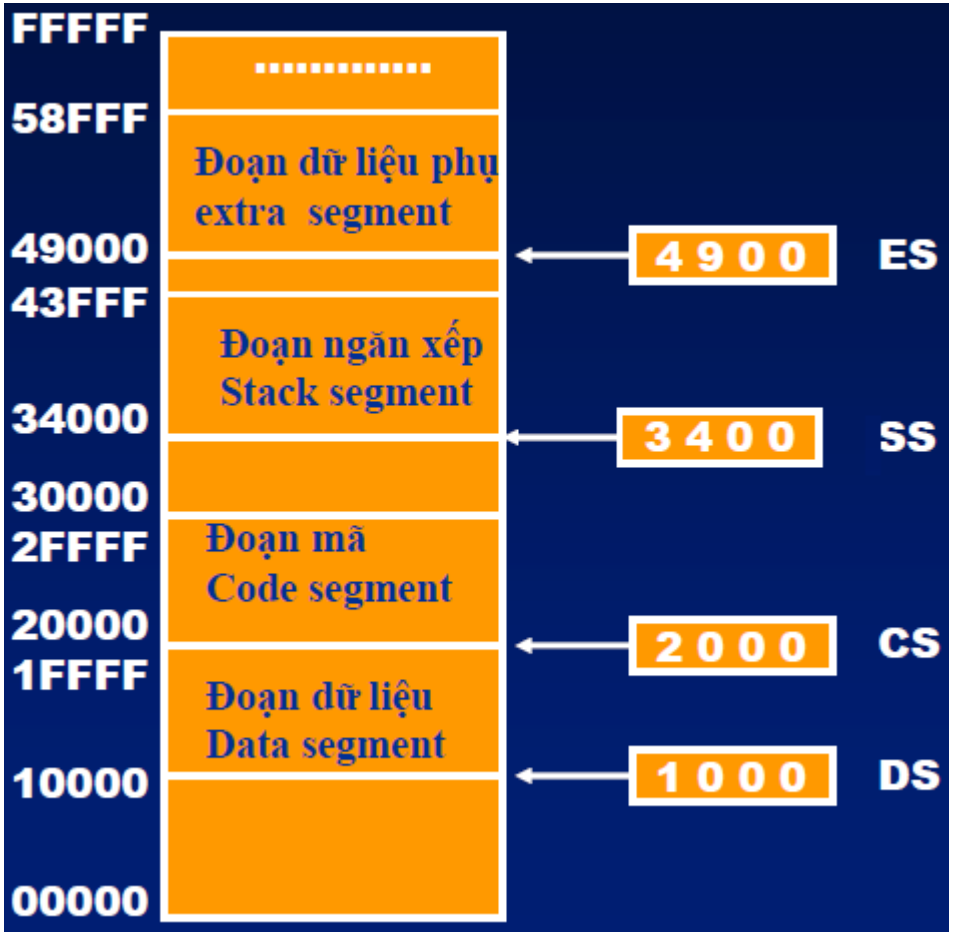
0300H*16=03000h

\+ 0F345h

12345h

Thanh ghi

- Các thanh ghi đoạn



Thanh ghi

- Các thanh ghi con trỏ và chỉ số
  - IP (Instruction Pointer): chứa địa chỉ lệnh tiếp theo sẽ được thực hiện (CS:IP)
  - BP (Base Pointer): chứa địa chỉ của dữ liệu trong đoạn ngăn xếp SS hoặc các đoạn khác (SS:BP)
  - SP (Stack Pointer): chứa địa chỉ hiện thời của đỉnh ngăn xếp (SS:SP)
  - SI (Source Index): chứa địa chỉ dữ liệu nguồn (DS:SI)
  - DI (Destination Index): chứa địa chỉ dữ liệu đích (ES:DI)

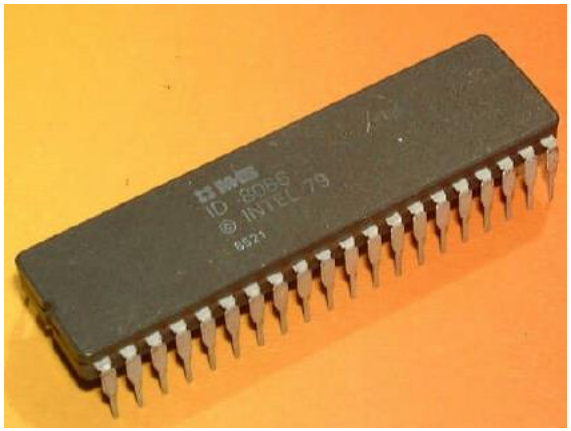
Thanh ghi

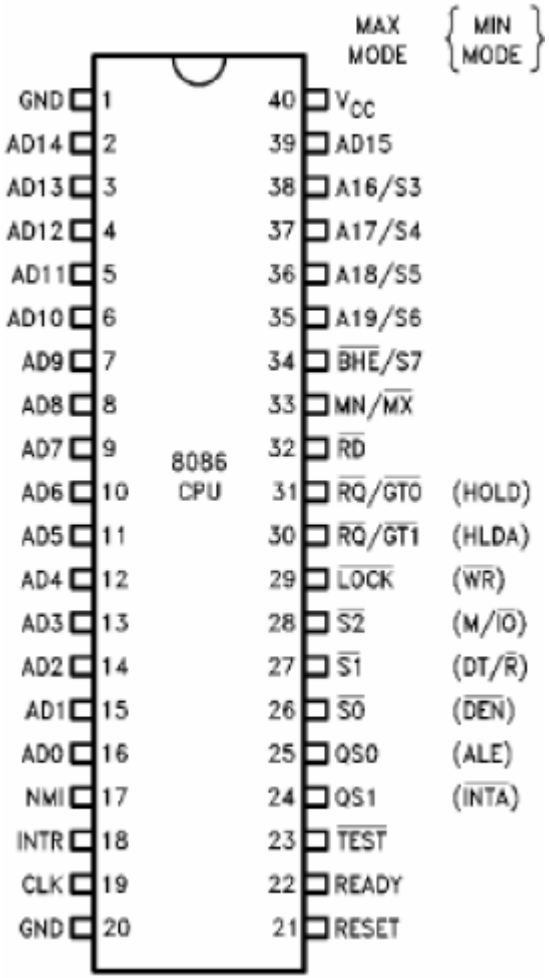
- Thanh ghi cờ (Flag)

  - C (Carry): CF=1 khi có nhớ hoặc mượn từ MSB
  - P (Parity): PF=0 (1) khi tổng số bít 1 trong kết quả là chẵn (lẻ)
  - A (Auxilary carry): AF=1 khi có nhớ hoặc mượn từ một số BCD thấp sang BCD cao
  - Z (Zero): ZF=1 khi kết quả bằng 0
  - S (Sign): SF=1 khi kết quả âm
  - O (Overflow): OF=1 khi kết quả là một số vượt ra ngoài giới hạn biểu diễn của nó trong khi thực hiện phép toán cộng trừ số có dấu
  - T (Trap): TF=1 khi CPU làm việc ở chế độ chạy từng lệnh
  - I (Interrupt enable): IF=1 khi CPU cho phép ngắt
  - D (Direction): DF=1 làm việc với chuỗi từ phải sang trái

Các chân tín hiệu

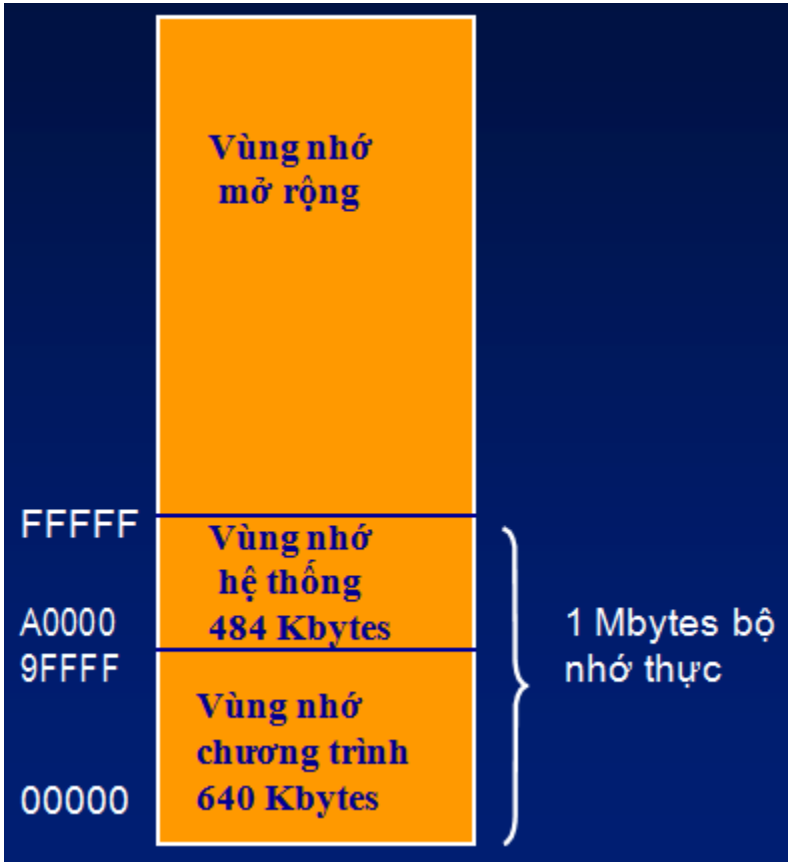
- Các chân tín hiệu của 8086





Bản đồ bộ nhớ

- Bản đồ bộ nhớ của 8086

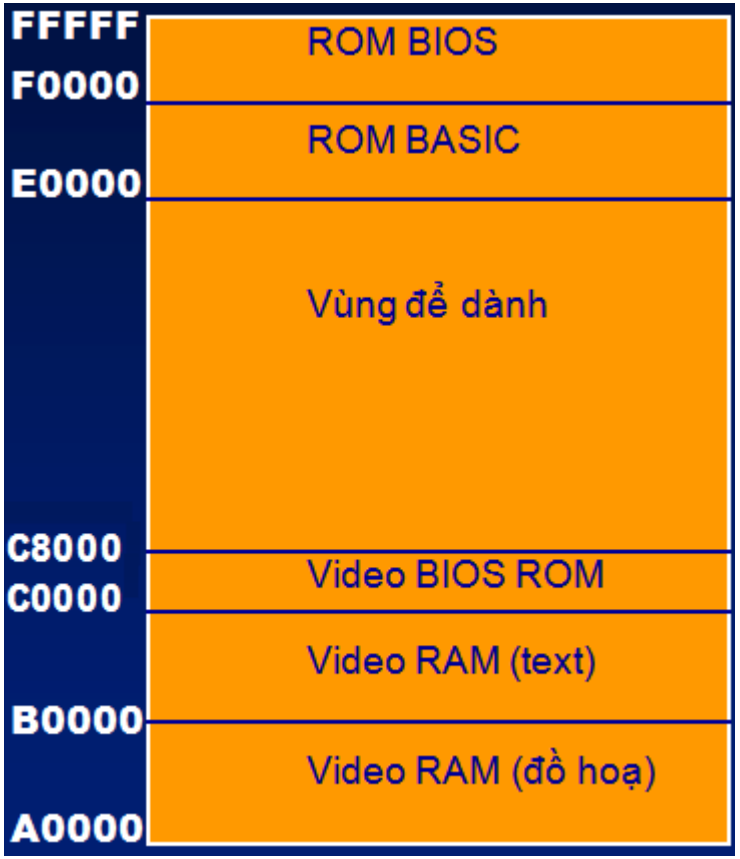


Bản đồ bộ nhớ

- Bản đồ vùng nhớ chương trình

Bản đồ bộ nhớ

- Bản đồ vùng nhớ hệ thống



Bản đồ bộ nhớ

- Các cổng vào/ra

Các chế độ địa chỉ

- Chế độ địa chỉ: cơ chế xác định toán hạng
  - Chế độ địa chỉ thanh ghi
  - Chế độ địa chỉ tức thì
  - Chế độ địa chỉ trực tiếp
  - Chế độ địa chỉ gián tiếp qua thanh ghi
  - Chế độ địa chỉ tương đối cơ sở
  - Chế độ địa chỉ tương đối chỉ số
  - Chế độ địa chỉ tương đối chỉ số cơ sở

Các chế độ địa chỉ

- Chế độ địa chỉ thanh ghi (Register Addressing Mode)
  - Toán hạng là các thanh ghi
  - Tốc độ thực hiện lệnh cao
- Ví dụ:

  MOV BX,DX ; BX ⬅ DX

  ADD AL,DL ; AL ⬅ AL+DL

Các chế độ địa chỉ

- Chế độ địa chỉ tức thì (Immediate Addressing Mode)
  - Toán hạng Đích là thanh ghi hoặc ô nhớ
  - Toán hạng Nguồn là hằng số
- Ví dụ:

  MOV AH,1 ; AH ⬅ 1

  MOV [BX],10 ; chuyển 10 vào ô nhớ có địa chỉ trong thanh ghi BX

  MOV BX,10 ;chuyển 10 vào thanh ghi BX

Các chế độ địa chỉ

- Chế độ địa chỉ trực tiếp (Direct Addressing Mode)
  - Một toán hạng là địa chỉ ô nhớ
  - Toán hạng còn lại là thanh ghi
- Ví dụ:

  MOV AL,[1234h]; AL ⬅ [1234h]

  MOV [1234h],DX ; [1234h] ⬅ DX

Các chế độ địa chỉ

- Chế độ địa chỉ gián tiếp qua thanh ghi (Register Indirect Addressing Mode)
  - Một toán hạng là thanh ghi chứa địa chỉ của 1 ô nhớ dữ liệu
  - Toán hạng còn lại là thanh ghi
- Ví dụ:

  MOV AL,[BX]; AL⬅ [BX]

  MOV [SI],DL ; [SI]⬅DL

Các chế độ địa chỉ

- Chế độ địa chỉ tương đối cơ sở (Based Relative Addressing Mode)
  - Một toán hạng là thanh ghi cơ sở (BX, BP) và các hằng số biểu diễn giá trị dịch chuyển
  - Toán hạng còn lại là thanh ghi
- Ví dụ:

  MOV AL,[BX]+10; AL ⬅ [BX] + 10

  MOV [BP+5],DL ; [BP]+5 ⬅ DL

Các chế độ địa chỉ

- Chế độ địa chỉ tương đối chỉ số (Indexed Relative Addressing Mode)
  - Một toán hạng là thanh ghi chỉ số (SI, DI) và các hằng số biểu diễn giá trị dịch chuyển
  - Toán hạng còn lại là thanh ghi
- Ví dụ:

  MOV AL,[SI]+10; AL⬅ [SI] + 10

  MOV [DI+5],DL ; [DI]+5 ⬅DL

Các chế độ địa chỉ

- Chế độ địa chỉ tương đối chỉ số (Indexed Relative Addressing Mode)
  - Một toán hạng là thanh ghi chỉ số (SI, DI) và các hằng số biểu diễn giá trị dịch chuyển
  - Toán hạng còn lại là thanh ghi
- Ví dụ:

  MOV AL,[SI]+10; AL⬅ [SI] + 10

  MOV [DI+5],DL ; [DI]+5 ⬅DL

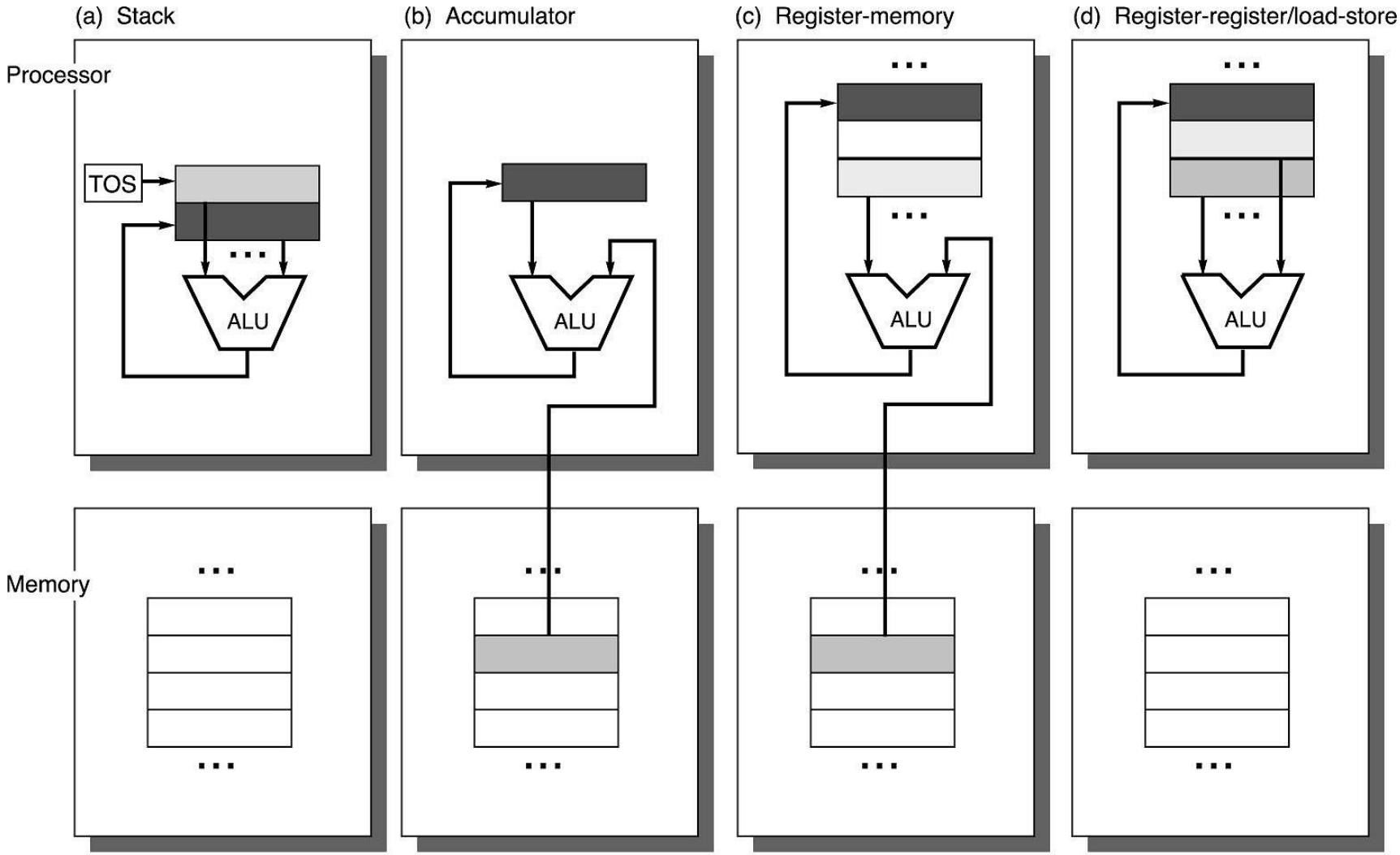
Các chế độ địa chỉ

- Chế độ địa chỉ tương đối chỉ số cơ sở (Based Indexed Relative Addressing Mode)
  - Một toán hạng là thanh ghi chỉ số (SI, DI), thanh ghi cơ sở và các hằng số biểu diễn giá trị dịch chuyển
  - Toán hạng còn lại là thanh ghi
- Ví dụ:

MOV AL,[BX][SI]+10; AL ⬅ [BX][SI] + 10

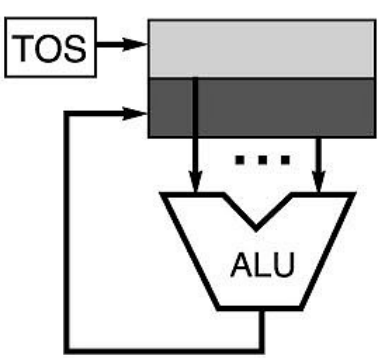
MOV [BP][DI+5],DL ; [BP][DI]+5 ⬅ DL

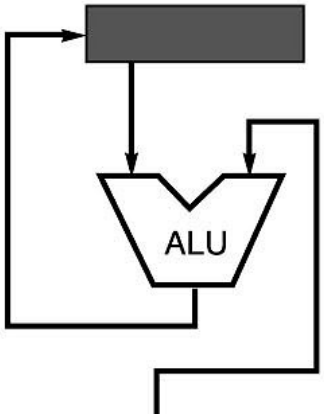
Các chế độ địa chỉ



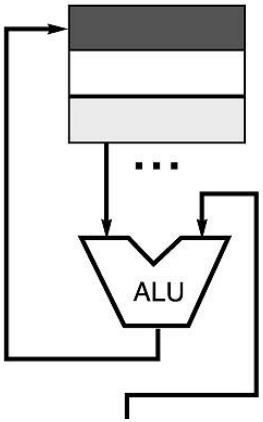
Các chế độ địa chỉ

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

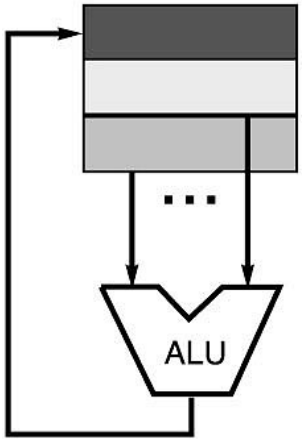




acc = acc + mem[C]



R1 = R1 + mem[C]



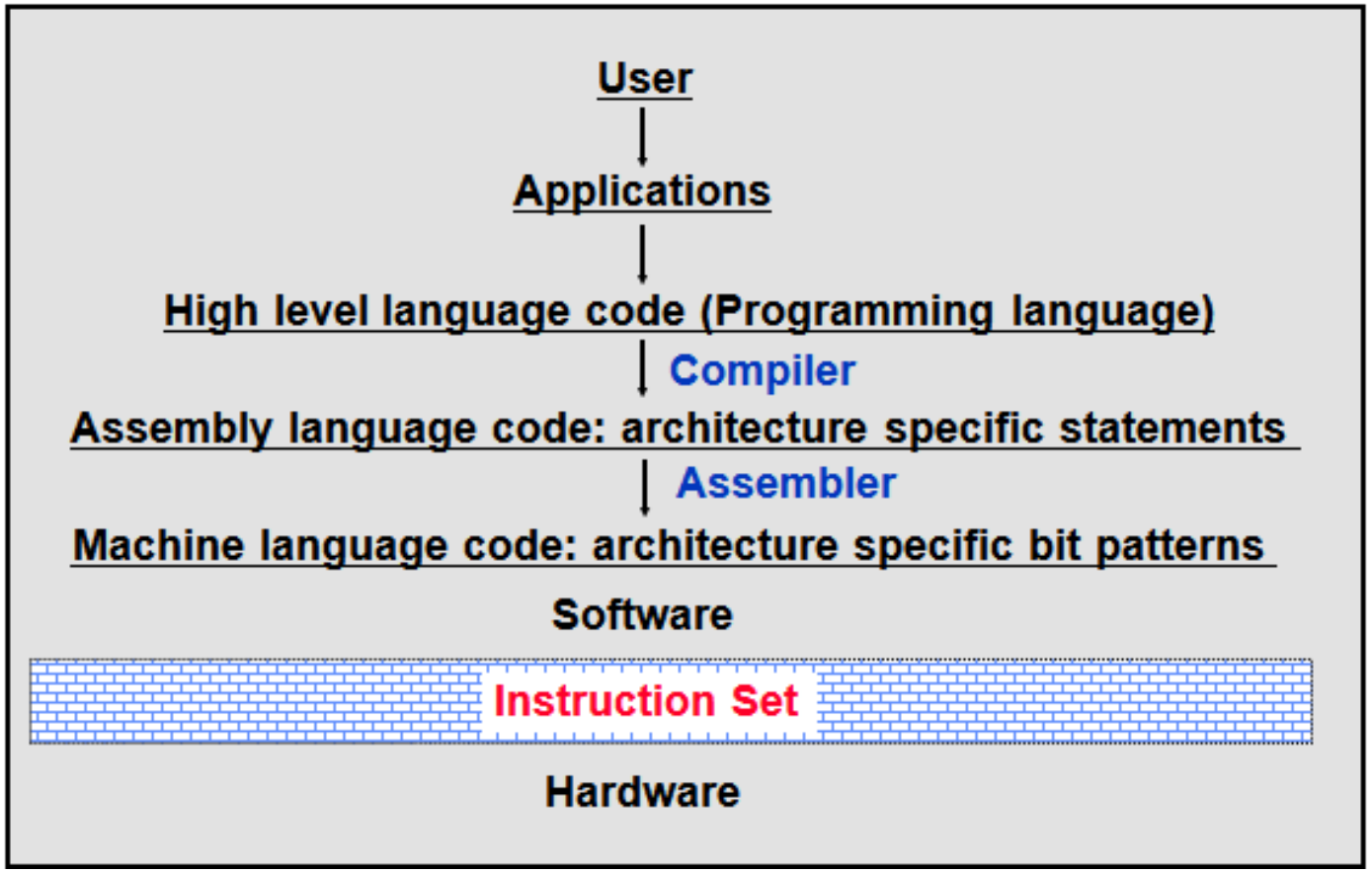
R3 = R1 + R2

Các chế độ địa chỉ

Chế độ địa chỉ	Toán hạng	Thanh ghi đoạn ngầm định
Thanh ghi	Thanh ghi	
Tức thì	Dữ liệu	
Trực tiếp	[offset]	DS
Gián tiếp qua thanh ghi	[BX] [SI] [DI]	DS DS DS
Tương đối cơ sở	[BX] + dịch chuyển [BP] + dịch chuyển	DS SS
Tương đối chỉ số	[DI] + dịch chuyển [SI] + dịch chuyển	DS DS
Tương đối chỉ số cơ sở	[BX] + [DI]+ dịch chuyển [BX] + [SI]+ dịch chuyển [BP] + [DI]+ dịch chuyển [BP] + [SI]+ dịch chuyển	DS DS SS SS

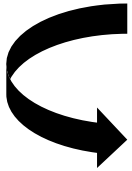
Cách mã hóa lệnh

- Giao tiếp người - máy



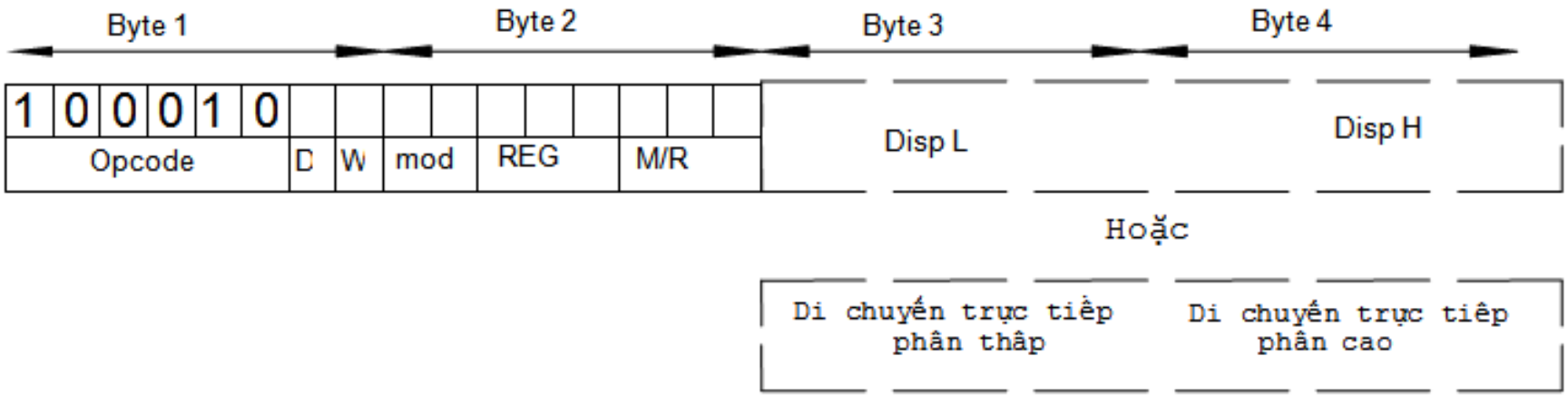
Cách mã hóa lệnh

- Ngôn ngữ
  - Ngôn ngữ lập trình: ngôn ngữ được thiết kế và chuẩn hóa để truyền các chỉ thị cho máy tính
  - Ngôn ngữ máy: dạng cơ bản nhất mà máy có thể hiểu được
  - Hợp ngữ: thay vì viết chương trình dưới dạng nhị phân sẽ dùng kí hiệu tượng trưng



Mã hóa lệnh: Hợp ngữ sang Mã máy

Cách mã hóa lệnh



- Opcode: mã lệnh, với lệnh MOV, Opcode = 100010
- D: đích đến của dữ liệu, D = 1 dữ liệu tới thanh ghi REG
- W: khuôn dạng dữ liệu, W = 0 với kiểu byte, 1 với kiểu word
- Mod/MR: chế độ địa chỉ của Toán hạng
- REG: mã thanh ghi

Cách mã hóa lệnh

Thanh ghi		Mã
W = 1	W = 0	
AX	AL	000
BX	BL	011
CX	CL	001
DX	DL	010
SP	AH	100
DI	BH	111
BP	CH	101
SI	DH	110

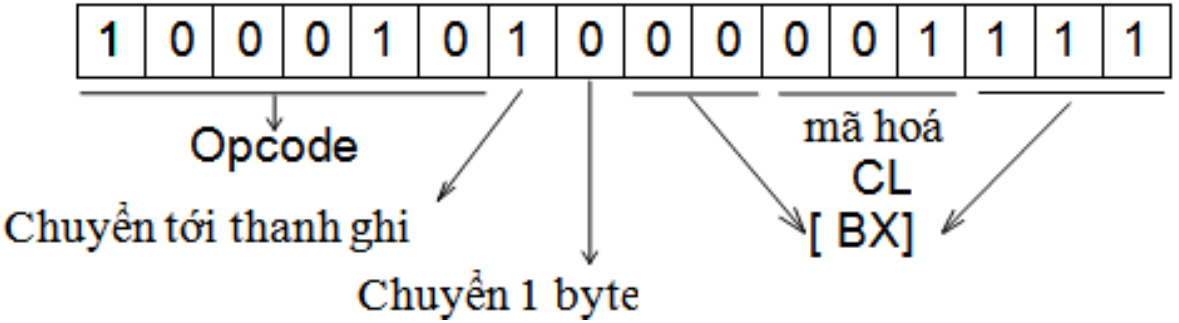
Thanh ghi đoạn	Mã
CS	01
DS	11
ES	00
SS	10

MOD / R/M	00	01	10	11	
				W=0	W=1
000	[BX]+[8]	[BX]+[SI]+d8	[BX]+[SI]+d16	AL	AX
001	[BX]+[DI]	[BX]+[DI]+d8	[BX]+[DI]+d16	CL	CX
010	[BP]+[SI]	[BP]+[SI]+d8	[BP]+[SI]+d16	DL	DX
011	[BP]+[DI]	[BP]+[DI]+d8	[BP]+[DI]+d16	BL	BX
100	[SI]	[SI]+d8	[SI]+d16	AH	SP
101	[DI]	[DI]+d8	[DI] +d16	CH	BP
110	d16	[BP]+d8	[BP]+d16	DH	SI
111	[BX]	[BX]+d8	[BX]+d16	BH	DI

Cách mã hóa lệnh

Ví dụ 1: mã hóa lệnh MOV CL,[BX]

- Chức năng: chuyển dữ liệu từ ô nhớ trỏ bởi BX sang thanh ghi CL
- D = 1: đích đến của dữ liệu là thanh ghi
- W = 0: dữ liệu kiểu byte
- Mod/MR = 00|111
- REG = 001



Mã của lệnh: 8A0Fh

Cách mã hóa lệnh

Ví dụ 1: mã hóa lệnh : MOV CH,[BX]

- Chức năng: chuyển dữ liệu từ ô nhớ trỏ bởi BX sang thanh ghi CH
- D = 1: đích đến của dữ liệu là thanh ghi
- W = 0: dữ liệu kiểu byte (CH: 8bit)
- Mod/MR = 00|111
- REG = 101

Mã: 100010 1 0 00 101 111

8A2Fh

Cách mã hóa lệnh

Ví dụ 2: mã hóa lệnh : MOV [BX]+0A285h,DX

- Chức năng: chuyển dữ liệu từ thanh ghi DX sang ô nhớ có địa chỉ [BX]+0A285h
- D = 0: đích đến của dữ liệu là ô nhớ
- W = 1: dữ liệu kiểu word (DX: 16bit)
- Mod/MR = 10|111
- REG = 010

Mã: 100010 0 1 10 010 111

899785A2h

Cách mã hóa lệnh

Ví dụ 3: mã hóa lệnh : MOV [BP]+[DI]+0F83H,AX

- Chức năng: chuyển dữ liệu từ thanh ghi AX sang ô nhớ có địa chỉ [BP]+[DI]+0F83H
- D = 0: đích đến của dữ liệu là ô nhớ
- W = 1: dữ liệu kiểu word (AX: 16bit)
- Mod/MR = 10|011
- REG = 000

Mã: 100010 0 1 10 000 011

8983830Fh

Cách mã hóa lệnh

Ví dụ 4: mã hóa lệnh : MOV [BX]+[SI]+0A74H,DX

Chức năng: chuyển dữ liệu từ thanh ghi DX sang ô nhớ có địa chỉ [BX]+[SI]+0A74H

- D = 0: đích đến của dữ liệu là ô nhớ
- W = 1: dữ liệu kiểu word (DX: 16bit)
- Mod/MR = 10|000
- REG = 010

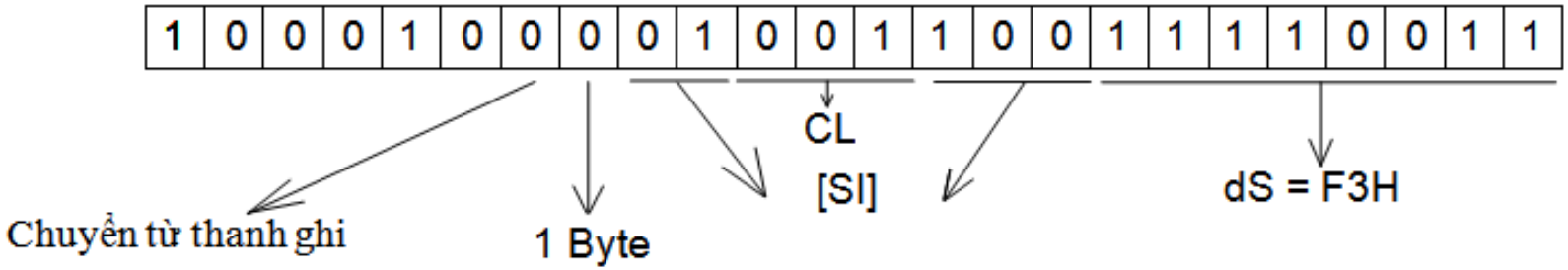
Mã: 100010 0 1 10 010 000

8990740Ah

Cách mã hóa lệnh

Ví dụ 2: mã hóa lệnh MOV 0F3h[SI],CL

- Chức năng: chuyển dữ liệu từ thanh ghi CL sang ô nhớ có địa chỉ [SI]+F3h
- D = 0: đích đến của dữ liệu là ô nhớ
- W = 0: dữ liệu kiểu byte
- Mod/MR = 01|100
- REG = 001



Mã của lệnh: 884CF3h

Cách mã hóa lệnh

Ví dụ 3: giải mã lệnh

Cho các lệnh có mã: 8BC8h, 8AF1h, 882Fh

- Giải mã các lệnh trên
- Cho các giá trị: AX= 832Bh, BX=229Ch, DS=542Ah. Xác định giá trị ô nhớ có địa chỉ vậy lý 5653Ch khi CPU thực hiện liên tiếp 3 lệnh trên?

8BC8h:

1000101111001000

D=1: thanh ghi

W=1: word

001: t/g CX

11000: AX

Lệnh: MOV CX,AX

Cách mã hóa lệnh

Ví dụ 3: giải mã lệnh

Cho các lệnh có mã: 8BC8h, 8AF1h, 882Fh

- Giải mã các lệnh trên
- Cho các giá trị: AX= 832Bh, BX=229Ch, DS=542Ah. Xác định giá trị ô nhớ có địa chỉ vậy lý 5653Ch khi CPU thực hiện liên tiếp 3 lệnh trên?

8AF1h :

1000101011110001

D=1: thanh ghi

W=0: byte

110: t/g DH

11001: CL

Lệnh: MOV DH,CL

Cách mã hóa lệnh

Ví dụ 3: giải mã lệnh

Cho các lệnh có mã: 8BC8h, 8AF1h, 882Fh

- Giải mã các lệnh trên
- Cho các giá trị: AX= 832Bh, BX=229Ch, DS=542Ah. Xác định giá trị ô nhớ có địa chỉ vậy lý 5653Ch khi CPU thực hiện liên tiếp 3 lệnh trên?

882Fh :

100010000101111

D=0: ô nhớ

W=0: byte

101: t/g CH

00111: [BX]

Lệnh: MOV [BX],CH

Cách mã hóa lệnh

Ví dụ 3: giải mã lệnh

Cho các lệnh có mã: 8BC8h, 8AF1h, 882Fh

- Giải mã các lệnh trên: MOV CX,AX; MOV DH,CL; MOV [BX],CH
- Cho các giá trị: AX= 832Bh, BX=229Ch, DS=542Ah. Xác định giá trị ô nhớ có địa chỉ vậy lý 5653Ch khi CPU thực hiện liên tiếp 3 lệnh trên?

Sau lệnh 1: CX= AX = 832Bh-> CH=83h, CL=2Bh

Sau lệnh 2: DH= CL = 2Bh-> DH=2Bh

Sau lệnh 3: [BX]= CH = 83h

Xác định ô nhớ [BX]: đc vật lí:

DS:BX = DS*16 + BX

DS*16 = 542Ah * 16 = 542A0h

DS*16 + BX = 542A0h + 229Ch = 5653Ch

Vậy ô nhớ có đc vật lí 5653Ch có giá trị 83h

Cách mã hóa lệnh

Ví dụ 4: Cho các giá trị: AL= 52h, BX=2243h, DS=5BA8h. Anh (Chị) hãy xác định giá trị cho ô nhớ có địa chỉ vậy lý 67F00h sau khi CPU thực hiện 3 lệnh liên tiếp: 8AF0h, 8AE6h, 88B73DA2h.

Lệnh 1: 8AF0h

100010 1 0 11 110 000

D=1: thanh ghi

W=0: byte

REG: 110 (DH)

Mod/RM: 11000 (AL)

MOV DH,AL

Cách mã hóa lệnh

Ví dụ 4: Cho các giá trị: AL= 52h, BX=2243h, DS=5BA8h. Anh (Chị) hãy xác định giá trị cho ô nhớ có địa chỉ vậy lý 67F00h sau khi CPU thực hiện 3 lệnh liên tiếp: 8AF0h, 8AE6h, 88B73DA2h.

Lệnh 2: 8AE6h

100010 1 0 11 100 110

D=1: thanh ghi

W=0: byte

REG: 100 (AH)

Mod/RM: 11110 (DH)

MOV AH,DH

Cách mã hóa lệnh

Ví dụ 4: Cho các giá trị: AL= 52h, BX=2243h, DS=5BA8h. Anh (Chị) hãy xác định giá trị cho ô nhớ có địa chỉ vậy lý 67F00h sau khi CPU thực hiện 3 lệnh liên tiếp: 8AF0h, 8AE6h, 88B73DA2h.

Lệnh 3: 88B7h

100010 0 0 10 110 111

D=0: ô nhớ

W=0: byte

REG: 110 (DH)

Mod/RM: 10111 ([BX]+d16)

MOV [BX]+0A23Dh,DH

Cách mã hóa lệnh

Ví dụ 4: Cho các giá trị: AL= 52h, BX=2243h, DS=5BA8h. Anh (Chị) hãy xác định giá trị cho ô nhớ có địa chỉ vậy lý 67F00h sau khi CPU thực hiện 3 lệnh liên tiếp: 8AF0h, 8AE6h, 88B73DA2h.

Lệnh 1: MOV DH,AL->DH=AL=52h

Lệnh 2: MOV AH,DH ->AH=DH=52h

Lệnh 3: MOV [BX]+0A23Dh,DH -> ô nhớ có giá trị 52h

Đc: DS:BX = DS*16 + BX + 0A23Dh

= 5BA8h*16 + 2243h + 0A23Dh

= 5BA80h + 2243h + 0A23Dh

= 5DCC3h + 0A23Dh = 67F00h

Cách mã hóa lệnh

Ví dụ 5: Cho các lệnh có mã: 8AE2h, 8AF4h, 899785A2h.

- Giải mã các lệnh trên
- DX= 582Eh, BX=15D3h, DS=32EFh. Xác định giá trị ô nhớ có địa chỉ vậy lý 3E748h?

8AE2h :

1000101011100010

D=1: t/g

W=0: byte

100: t/g AH

11010: DL

Lệnh: MOV AH,DL

Cách mã hóa lệnh

Ví dụ 5: Cho các lệnh có mã: 8AE2h, 8AF4h, 899785A2h.

- Giải mã các lệnh trên
- DX= 582Eh, BX=15D3h, DS=32EFh. Xác định giá trị ô nhớ có địa chỉ vậy lý 3E748h?

8AF4h :

10001010111110100

D=1: t/g

W=0: byte

110: t/g DH

11010: AH

Lệnh: MOV DH,AH

Cách mã hóa lệnh

Ví dụ 5: Cho các lệnh có mã: 8AE2h, 8AF4h, 899785A2h.

- Giải mã các lệnh trên
- DX= 582Eh, BX=15D3h, DS=32EFh. Xác định giá trị ô nhớ có địa chỉ vậy lý 3E748h?

899785A2h :

1000100110010111

D=0: ô nhớ

W=1: word

010: t/g DX

10111: [BX] + d16

Lệnh: MOV [BX]+0A285h,DX

Cách mã hóa lệnh

Ví dụ 5: Cho các lệnh có mã: 8AE2h, 8AF4h, 899785A2h.

Giải mã các lệnh trên: MOV AH,DL; MOV DH,AH; MOV [BX]0A285h,DX

- DX= 582Eh, BX=15D3h, DS=32EFh. Xác định giá trị ô nhớ có địa chỉ vậy lý 3E748h?

Sau lệnh 1: AH= DL = 2Eh (DX=582Eh->DH=58h,DL=2Eh)

Sau lệnh 2: DH= AH = 2Eh -> DX=DH,DL= 2E2Eh

Sau lệnh 3: [BX]+0A285h = DX = 2E2Eh

Xác định ô nhớ [BX]+ 0A285h: đc vật lí:

DS:BX = DS*16 + BX + 0A285h

DS*16 = 32EFh * 16 = 32EF0h

DS*16 + BX = 32EF0h + 15D3h = 344C3h

DS*16 + BX + 0A285h = 344C3h + 0A285h = 3E748h

Vậy ô nhớ có đc vật lí 3E748h có giá trị 2E2Eh

Cách mã hóa lệnh

Ví dụ 5: Cho các lệnh có mã: 8BD9h, 8BC3h, 8886AD36h.

- Giải mã các lệnh trên
- CX= F523h, BP=12A3h, SS=2E1Bh. Xác định giá trị ô nhớ có địa chỉ vậy lý 32B00h?

Tập lệnh của 8086

- Các nhóm lệnh
  - Nhóm lệnh vào/ra: IN, OUT, INT_N,..
  - Nhóm lệnh chuyển dữ liệu: MOV, XCHG,..
  - Nhóm lệnh số học: ADD, SUB, MUL, DIV, INC, DEC,..
  - Nhóm lệnh logic: AND, OR, NOT, XOR,..
  - Nhóm lệnh dịch, quay: SHL, SHR, ROL, ROR,..
  - Nhóm lệnh điều khiển, rẽ nhánh: CALL, RET, JMP,…
  - Nhóm lệnh với dấu phẩy động: ADDF, MULF, DIVF,...
  - Nhóm lệnh thao tác với chuỗi: LOSB, STOSB, COMPARE,…

Lập trình Hợp ngữ với 8088

- Tổng quan
- Tập lệnh
- Các cấu trúc lập trình cơ bản
- Thao tác với các hệ cơ số
- Mảng và chuỗi
- Lập trình hệ thống

Tổng quan

- Ngôn ngữ
  - Ngôn ngữ lập trình: ngôn ngữ được thiết kế và chuẩn hóa để truyền các chỉ thị cho máy tính
  - Ngôn ngữ máy: dạng cơ bản nhất mà máy có thể hiểu được
  - Hợp ngữ: thay vì viết chương trình dưới dạng nhị phân sẽ dùng kí hiệu tượng trưng
- Chương trình hợp ngữ (Assembly - ASM)
  - Các dòng lệnh
  - Lệnh: lệnh thật dưới dạng ký hiệu (symbolic), dẫn hướng chương trình dịch, …
  - Khuôn dạng: theo quy tắc cú pháp nhất định để chương trình dịch có thể hiểu được: .COM, .EXE

Khung chương trình

- Khung chương trình dịch ra đuôi EXE

```
.MODEL SMALL
.STACK 100
.DATA
        ; các định nghĩa cho biến và hằng
.CODE
MAIN PROC
        ; khởi tạo dữ liệu
          MOV AX,@Data
          MOV DS,AX
        ; các lệnh của chương trình chính
        ; trở về DOS dùng hàm 4CH của ngắt INT 21h
          MOV AH,4Ch
          INT 21h
MAIN ENDP
        ; các chương trình con (nếu có)
END MAIN
```

Khung chương trình

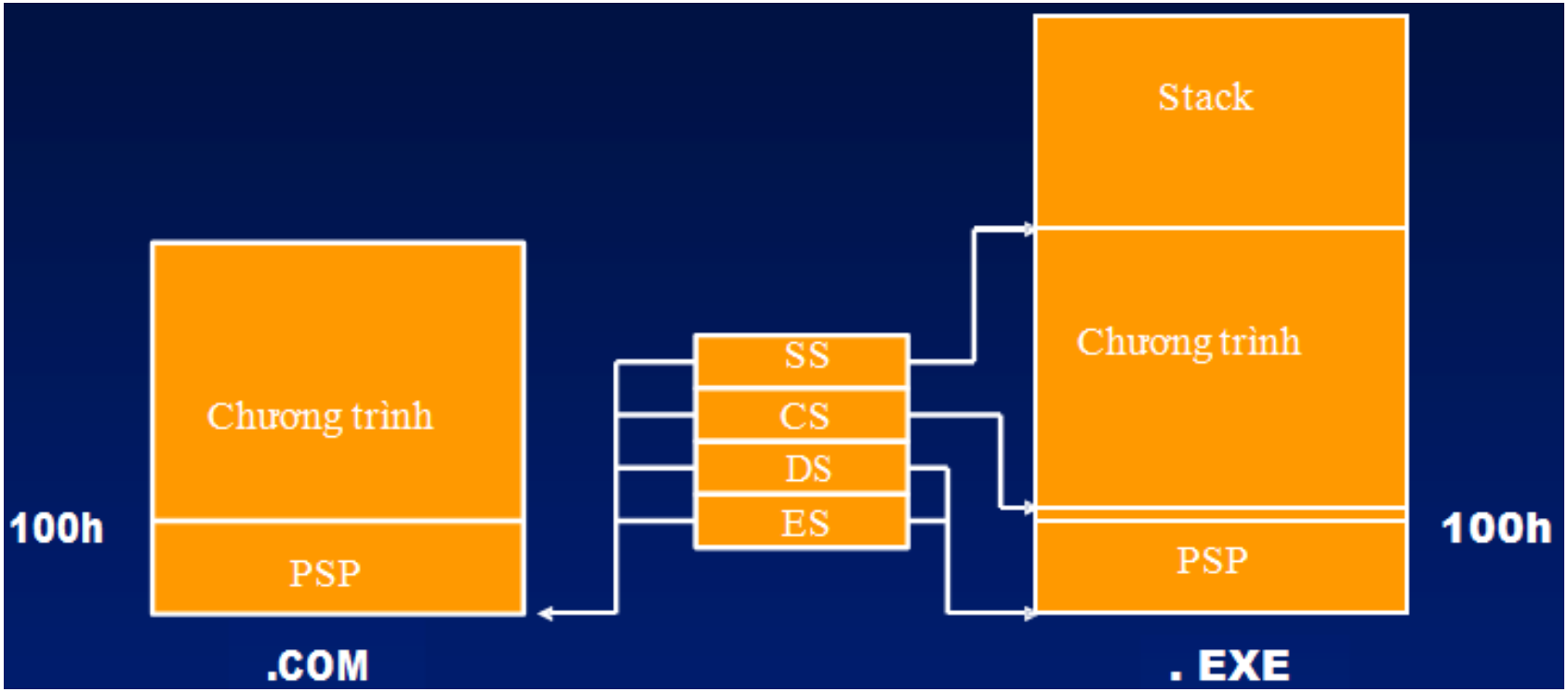
- Trong đó:
  - MODEL: khai báo qui mô sử dụng bộ nhớ (64Kb)
  - STACK: khai báo đoạn ngăn xếp (256byte)
  - DATA: đoạn dữ liệu
  - CODE: đoạn mã
  - MAIN PROC…MAIN ENDP: thân chương trình chính
  - END MAIN: kết thúc chương trình

Khung chương trình

- Khung chương trình dịch ra đuôi COM

```
.MODEL TINY
.CODE
        ORG 100h
START: JMP CONTINUE
        ; khai báo dữ liệu
CONTUNUE:
MAIN PROC
        ; các lệnh của chương trình chính
        ; trở về DOS dùng ngắt INT 20h
        INT 20h
MAIN ENDP
        ; các chương trình con (nếu có)
END START
```

Khung chương trình



Bản đồ bộ nhớ chương trình dịch ra đuôi .COM và .EXE

Khung chương trình

- Ví dụ 1: hiển thị lời chào “Hello”

```
.MODEL SMALL
.STACK 100
.DATA
        tb DB “Hello$”
.CODE
MAIN PROC
        MOV AX,@Data
        MOV DS,AX
        MOV AH,9
        LEA DX,tb
        INT 21h
        MOV AH,4Ch
        INT 21h
MAIN ENDP
END MAIN
```

Cấu trúc lệnh

- Dòng lệnh:

  Tên Mã lệnh Toán hạng ; chú giải

Tên:

- Không phân biệt chữ hoa, chữ thường
- Chứa các nhãn, tên biến, tên thủ tục
- Độ dài: 1 đến 31 ký tự
- Không được có dấu cách, không bắt đầu bằng số
- Được dùng các ký tự đặc biệt: ? . @ _$ %
- Dấu chấm (.) phải được đặt ở vị trí đầu tiên nếu sử dụng
- Nhãn kết thúc bằng dấu hai chấm (:)

Dữ liệu

- Dữ liệu:
  - Hệ số 2, ví dụ: 0011B
  - Hệ số 10, ví dụ: 1234
  - Hệ số 16, ví dụ: 1EF1H, 0ABBAH
  - Ký tự, chuỗi ký tự, ví dụ: ‘A’, ‘abcd’

Biến và hằng

- Kiểu dữ liệu:
  - DB (Define Byte): định nghĩa biến kiểu byte
  - DW (Define Word): định nghĩa biến kiểu từ
  - DD (Define Double Word): định nghĩa biến kiểu từ kép
- Biến kiểu byte:

  Tên DB Giá trị khởi gán
- Biến kiểu word:

  Tên DW Giá trị khởi gán
- Biến mảng:

  Tên Kiểu Giá trị khởi gán cho các phần tử

  hoặc

  Tên Kiểu Số phần tử DUP(giá trị khởi gán)

Biến và hằng

- Biến kiểu xâu kí tự:

  Tên Kiểu "Xâu kí tự"

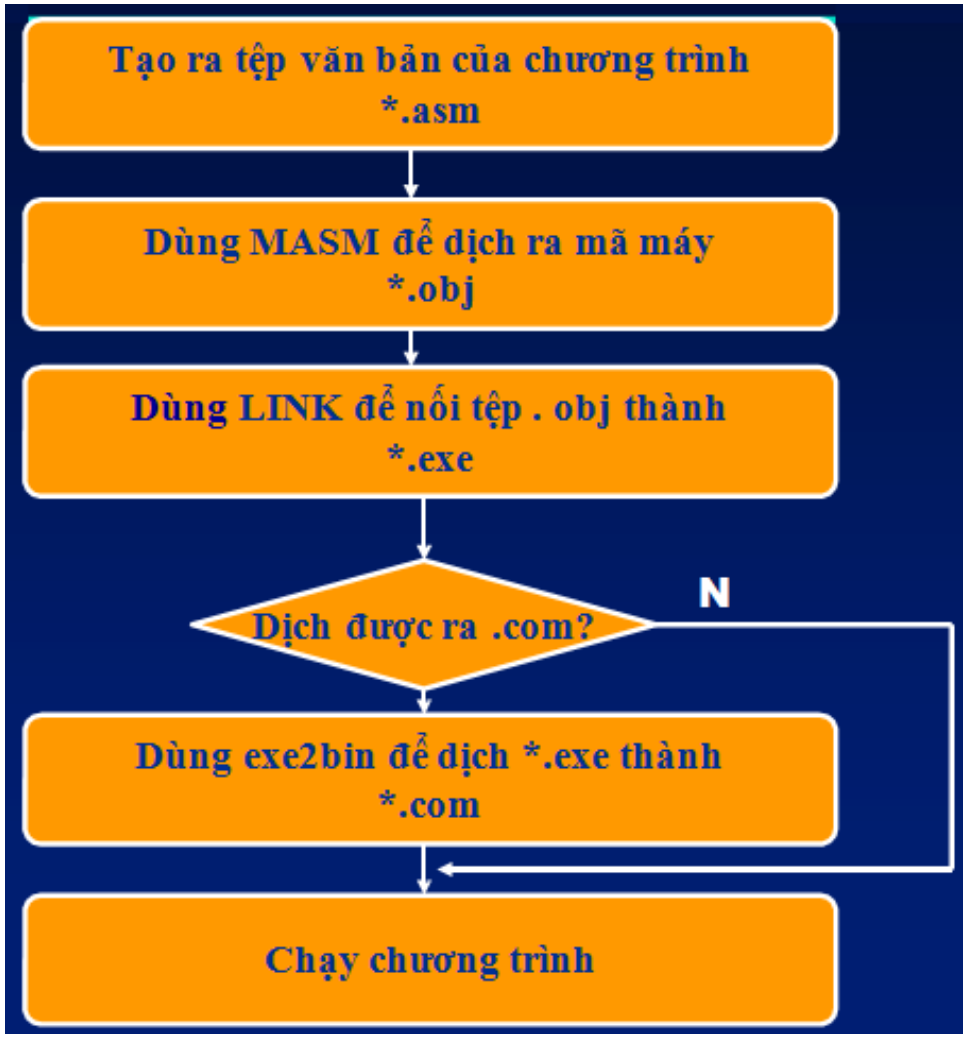
  hoặc

  Tên Kiểu Giá trị từng kí tự trong xâu

- Hằng có tên:

  Tên EQU Giá trị khởi gán

Biên tập và thực thi chương trình



Tập lệnh

- Các nhóm lệnh
  - Nhóm lệnh vào/ra: IN, OUT, INT_N,..
  - Nhóm lệnh chuyển dữ liệu: MOV, XCHG,..
  - Nhóm lệnh số học: ADD, SUB, MUL, DIV, INC, DEC,..
  - Nhóm lệnh logic: AND, OR, NOT, XOR,..
  - Nhóm lệnh dịch, quay: SHL, SHR, ROL, ROR,..
  - Nhóm lệnh điều khiển, rẽ nhánh: CALL, RET, JMP,…
  - Nhóm lệnh với dấu phẩy động: ADDF, MULF, DIVF,...
  - Nhóm lệnh thao tác với chuỗi: LOSB, STOSB, COMPARE,…

Tập lệnh

- Nhóm lệnh vào/ra: IN, OUT, INT_N,..
- Vào/ra trực tiếp: IN, OUT

IN Thanh ghi,Số hiệu cổng

- Chức năng: đọc dữ liệu từ 1 cổng nào đó vào thanh ghi
- Ví dụ:

  IN AL,80h; đọc dữ liệu từ cổng 80h vào thanh ghi AL

OUT Số hiệu cổng,Thanh ghi

- Chức năng: đưa dữ liệu từ 1 thanh ghi ra cổng nào đó
- Ví dụ:

  OUT 81h,DL; đưa dữ liệu từ thanh ghi DL ra cổng 81hs

Tập lệnh

- Nhóm lệnh vào/ra: IN, OUT, INT_N,..
- Vào/ra gián tiếp: dùng ngắt INT 21h

Hàm 1:

- Chức năng: nhập 1 kí tự từ bàn phím, mã ASCII của kí tự được đặt trong thanh ghi AL
- Ví dụ:

```
MOV AH,1
INT 21h
```

Hàm 2:

- Chức năng: hiển thị 1 kí tự ra màn hình, mã ASCII của kí tự cần hiển thị được đặt trong thanh ghi DL
- Ví dụ:

```
MOV AH,2
INT 21h
```

Tập lệnh

Với Hàm 2, khi giá trị DL là:

- 13 (phím ENTER): xuống dòng
- 10 (phím TAB): về đầu dòng

Hàm 9:

- Chức năng: hiển thị 1 chuỗi kí tự ra màn hình, địa chỉ của chuỗi cần hiển thị được đặt trong thanh ghi DX
- Ví dụ:

```
LEA DX,Str
MOV AH,9
INT 21h
```

Tập lệnh

- Nhóm lệnh chuyển dữ liệu: MOV (Move), XCHG (Exchange), PUSH, POP,..

MOV Đích,Nguồn

- Chức năng: chuyển dữ liệu từ 1 thanh ghi, 1 ô nhớ, 1 toán hạng trực tiếp sang 1 thanh ghi hay 1 ô nhớ
- Ví dụ:

  MOV AH,1; AH ⬅ 1

  MOV [SI],DX; [SI] ⬅ DX

XCHG Đích,Nguồn

- Chức năng: hoán chuyển nội dung của 2 thanh ghi, 1 thanh ghi và 1 ô nhớ
- Ví dụ:

  XCHG AX,BX; AX ⬅ BX; BX ⬅ AX

Tập lệnh

- Nhóm lệnh chuyển dữ liệu: MOV (Move), XCHG (Exchange), PUSH, POP,..

PUSH Nguồn

- Chức năng: cất 1 word từ 1 thanh ghi hay 1 ô nhớ vào stack
- Ví dụ:

  PUSH AX; TOS ⬅ AX

POP Đích

- Chức năng: lấy 1 phần tử đỉnh stack đưa vào 1 thanh ghi hay 1 ô nhớ
- Ví dụ:

  POP BX; BX ⬅ TOS

Tập lệnh

- Nhóm lệnh số học: ADD (Addition), SUB (Subtract), MUL (Multiple), DIV (Division),..

  ADD Đích,Nguồn

  - Chức năng: cộng nội dung của 2 thanh ghi, 1 thanh ghi và 1 ô nhớ hay 1 toán hạng trực tiếp
  - Ví dụ:

    ADD AX,BX; AX ⬅ AX + BX

  SUB Đích,Nguồn

  - Chức năng: trừ nội dung của 2 thanh ghi, 1 thanh ghi và 1 ô nhớ hay 1 toán hạng trực tiếp
  - Ví dụ:

    SUB BX,5; BX ⬅ BX - 5

Tập lệnh

- Nhóm lệnh số học: ADD (Addition), SUB (Subtract), MUL (Multiple), DIV (Division),..

  MUL Nguồn

  - Chức năng: nhân nội dung của 2 thanh ghi, 1 thanh ghi và 1 ô nhớ
  - Toán hạng ngầm định chứa trong AL (byte) hoặc AX (word); kết quả được lưu trong AX (DX:AX)
  - Ví dụ:

    MUL BL; AX ⟵ AL * BL

Tập lệnh

- Nhóm lệnh số học: ADD (Addition), SUB (Subtract), MUL (Multiple), DIV (Division),..

  DIV Nguồn

  - Chức năng: Chia nội dung của 2 thanh ghi, 1 thanh ghi và 1 ô nhớ
  - Số bị chia ngầm định đặt trong AX (byte) hoặc DX:AX (word);
  - Thương số được lưu trong AL (AX), số dư được lưu trong AH (DX)
  - Ví dụ:

    MOV AX,13

    MOV DL,5

    DIV DL; AL⬅ 2; AH ⬅3

Tập lệnh

- Nhóm lệnh số học: ADD (Addition), SUB (Subtract), MUL (Multiple), DIV (Division),..

  INC Toán hạng

  - Chức năng: tăng nội dung của 1 thanh ghi lên 1
  - Ví dụ:

    INC CX; CX⬅ CX + 1

  DEC Toán hạng

  - Chức năng: giảm nội dung của 1 thanh ghi đi 1
  - Ví dụ:

    DEC CX; CX⬅ CX - 1

Tập lệnh

- Nhóm lệnh logic: AND, OR, NEG, XOR..

AND Đích,Nguồn

- Chức năng: thực hiện phép Và giữa 2 thanh ghi, 1 thanh ghi và 1 ô nhớ hay toán hạng trực tiếp
- Xóa 1 hay nhiều bit trong 1 toán hạng
- Ví dụ:

  MOV AL,39h

  AND AL,0Fh; AL ⬅ 09h

OR Đích,Nguồn

- Chức năng: thực hiện phép Hoặc giữa 2 thanh ghi, 1 thanh ghi và 1 ô nhớ hay toán hạng trực tiếp
- Thiết lập 1 hay nhiều bit trong 1 toán hạng
- Ví dụ:

  MOV DL,09h

  OR DL,30h; DL ⬅ 39h

Tập lệnh

- Nhóm lệnh logic: AND, OR, NEG, XOR..

XOR Đích,Nguồn

- Chức năng: thực hiện phép cộng Modulo 2 giữa 2 thanh ghi
- Xóa nội dung 1 thanh ghi nào đó
- Ví dụ:

  XOR BX,BX; BX ⬅ 0

NEG Toán hạng

- Chức năng: thực hiện phép lấy phần bù (số bù 2) 1 thanh ghi, 1 ô nhớ
- Ví dụ:

  MOV DL,09h

  NEG DL; DL ⬅ F7h

Tập lệnh

- Nhóm lệnh logic: SHL (Shift Left), SHR, ROL (Rotate Left), ROR

  SHL Toán hạng,1

  SHL Toán hạng,CL

  - Chức năng: thực hiện phép dịch trái 1 toán hạng đi 1 hay nhiều vị trí (bit)

  - Thực hiện phép nhân với số 2^n
  - Ví dụ:

    MOV BX,2

    MOV CL,4

    SHL BX,CL; BX ⬅ 32

Tập lệnh

- Nhóm lệnh logic: SHL (Shift Left), SHR, ROL (Rotate Left), ROR

  SHR Toán hạng,1

  SHR Toán hạng,CL

  - Chức năng: thực hiện phép dịch phải 1 toán hạng đi 1 hay nhiều vị trí (bit)

  - Thực hiện phép chia với số 2^n
  - Ví dụ:

    MOV BX,32

    MOV CL,4

    SHR BX,CL; BX⬅ 2

Tập lệnh

- Nhóm lệnh logic: SHL (Shift Left), SHR, ROL (Rotate Left), ROR

  ROL Toán hạng,1

  ROL Toán hạng,CL

  - Chức năng: thực hiện phép quay trái 1 toán hạng đi 1 hay nhiều vị trí (bit) ~ kiểm soát từng bit trong toán hạng

  - Ví dụ:

    MOV BX,1234h

    MOV CL,4

    ROL BX,CL; BX⬅ 2341h

Tập lệnh

- Nhóm lệnh điều khiển, rẽ nhánh: JMP (Jump), JA (JG), JB (JL), JE (JZ),…

  JMP Nhãn đích

  - Chức năng: nhảy (không điều kiện) tới 1 địa chỉ trong bộ nhớ
  - Nhãn đích: ngắn (short: 2byte), gần (near:32Kbyte), xa (far: cho phép nhảy tới nhãn trong đoạn mã khác)
  - Ví dụ:

```
Lap:
        MOV AH,1
        INT 21h
        CMP AL,13
        JE Thoat
        MOV [SI],AL
        INC CX
        JMP Lap
```

Tập lệnh

- Nhóm lệnh điều khiển, rẽ nhánh: JMP (Jump), JA (JG), JB (JL), JE (JZ),…

  JA Nhãn đích

  - Chức năng: nhảy nếu lớn hơn (có điều kiện) tới 1 địa chỉ trong bộ nhớ
  - Điều kiện của lệnh được kiểm tra bởi lệnh CMP (Compare)
  - Ví dụ:

```
Lap:
        MOV AH,1
        INT 21h
        CMP AL,13
        JE Thoat
        CMP AL,30h
        JB Loi
        CMP AL,39h
        JA Loi
```

Tập lệnh

- Nhóm lệnh điều khiển, rẽ nhánh: LOOP, CALL, RET,...

LOOP Nhãn đích

- Chức năng: thực hiện 1 chu trình (vòng lặp có số lần lặp biết trước)
- Số lần thực hiện của chu trình (lặp) được đặt trong CX
- CX tự động giảm 1 cho tới khi kết thúc chu trình (CX = 0)
- Ví dụ: hiển thị 5 dấu '*'

```
        MOV CX,5
        MOV AH,2
Lap:
        MOV DL,'*'
        INT 21h
        LOOP Lap
```

Tập lệnh

- Nhóm lệnh điều khiển, rẽ nhánh: LOOP, CALL, RET,...

CALL Tên chương trình

- Chức năng: gọi chương trình con
- Chương trình con phải được khai báo hoặc chỉ rõ đường dẫn tới tệp (sau lệnh MAIN ENDP)
- Ví dụ:

  C

RET

- Chức năng: trở về chương trình chính
- Kết thúc chương trình con

Tập lệnh

- Nhóm lệnh thao tác với chuỗi: CLD, STD, LODSB(W), STOSB(W),..

CLD

- Chức năng: thực hiện thao tác từ phải qua trái

STD

- Chức năng: thực hiện thao tác từ trái qua phải

LODSB(W)

- Chức năng: chuyển nội dung phần tử trong chuỗi hiện thời trỏ bởi SI vào thanh ghi AL (SI tự động tăng/giảm 1/2)

STOSB(W)

- Chức năng: chuyển nội dung thanh ghi AL vào phần tử trong chuỗi hiện thời trỏ bởi DI (DI tự động tăng/giảm 1/2)

Các cấu trúc lập trình cơ bản

- Cấu trúc IF…THEN

Ví dụ: gán cho BX giá trị tuyệt đối của AX

- Nếu AX >= 0 thì BX = AX
- Nếu AX < 0 thì BX = - AX

ASM:

```
    CMP AX,0; so sanh AX voi 0
    JB Am; nếu nhỏ hơn thì chuyển tới nhãn Am
    MOV BX,AX
    JMP Thoat
Am:
    NEG AX
    MOV BX,AX
Thoat:
```

Các cấu trúc lập trình cơ bản

- Cấu trúc CASE

Ví dụ: gán cho BX giá trị:

- Nếu AX > 0 thì BX = 1
- Nếu AX = 0 thì BX = 0
- Nếu AX < 0 thì BX = - 1

ASM:

```
        CMP AX,0
        JA Duong
        JE Khong
        JB Am
        JMP Thoat
Duong:
        MOV BX,1
        JMP Thoat
Khong:
        MOV BX,0
        JMP Thoat
Am:
        MOV BX,-1
Thoat:
```

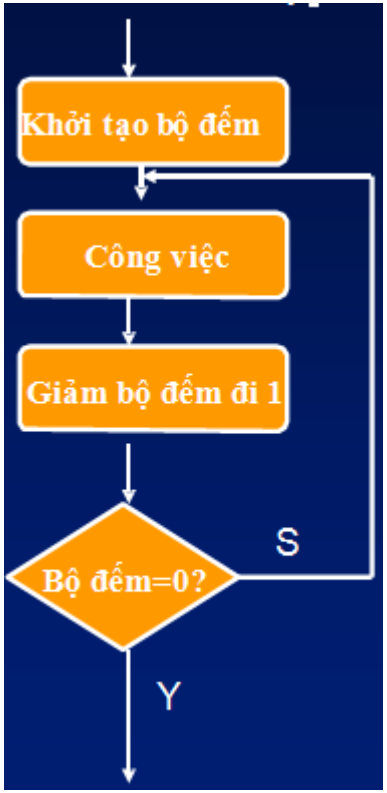
Các cấu trúc lập trình cơ bản

- Cấu trúc FOR

Ví dụ: hiển thị ra màn hình 5 dấu *

ASM:

```
    MOV CX,5
    MOV AH,2
    MOV DL,'*'
Lap:
    INT 21h
    LOOP Lap
```

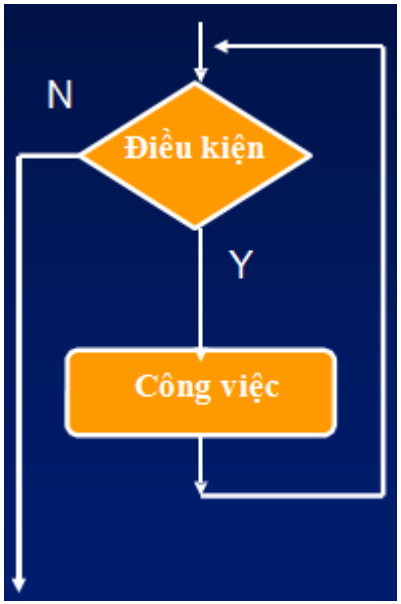


Các cấu trúc lập trình cơ bản

- Cấu trúc WHILE

Ví dụ: nhập kí tự từ bàn phím, kết thúc nhập bởi phím ENTER

```
ASM:
        MOV AH,1
Lap:
        INT 21h
        CMP AL,13
        JE Thoat
        …
```



Thao tác với các hệ cơ số

- Nhập xuất số hệ 2
- Nhập xuất số hệ 16
- Nhập xuất số hệ 10
- Mảng dữ liệu

Thao tác với các hệ cơ số

- Nhập số hệ 2
  - Số hệ 2: 0,1 ➡ kiểm tra tính đúng đắn của dữ liệu (so sánh với 30h, 31h)
  - Chuyển sang giá trị thực
  - Lưu trữ: dịch trái BX 1 vị trí và chèn bit mới nhập vào LSB
  - Tăng biến đếm (số bit được nhập)
- Xuất số hệ 2
  - Quay trái BX 1 vị trí, nếu có nhớ ➡ hiện 1, ngược lại hiện 0
  - Lặp lại

Thao tác với các hệ cơ số

- Nhập số hệ 16
  - Số hệ 16: chữ số 0,..9, chữ cái 'A'…'F' ➡ kiểm tra tính đúng đắn của dữ liệu (so sánh với 30h…39h và từ 'A'…'F')
  - Chuyển sang giá trị thực (trừ 30h nếu là chữ số, 37h nếu là chữ cái)
  - Lưu trữ: dịch trái BX 4 vị trí và chèn số mới nhập vào vị trí trái nhất
  - Tăng biến đếm (số bit được nhập)
- Xuất số hệ 16
  - Quay trái BX 4 vị trí
  - So sánh với 9, nhỏ hơn ➡hiện chữ số, ngược lại chữ cái
  - Lặp lại

Thao tác với các hệ cơ số

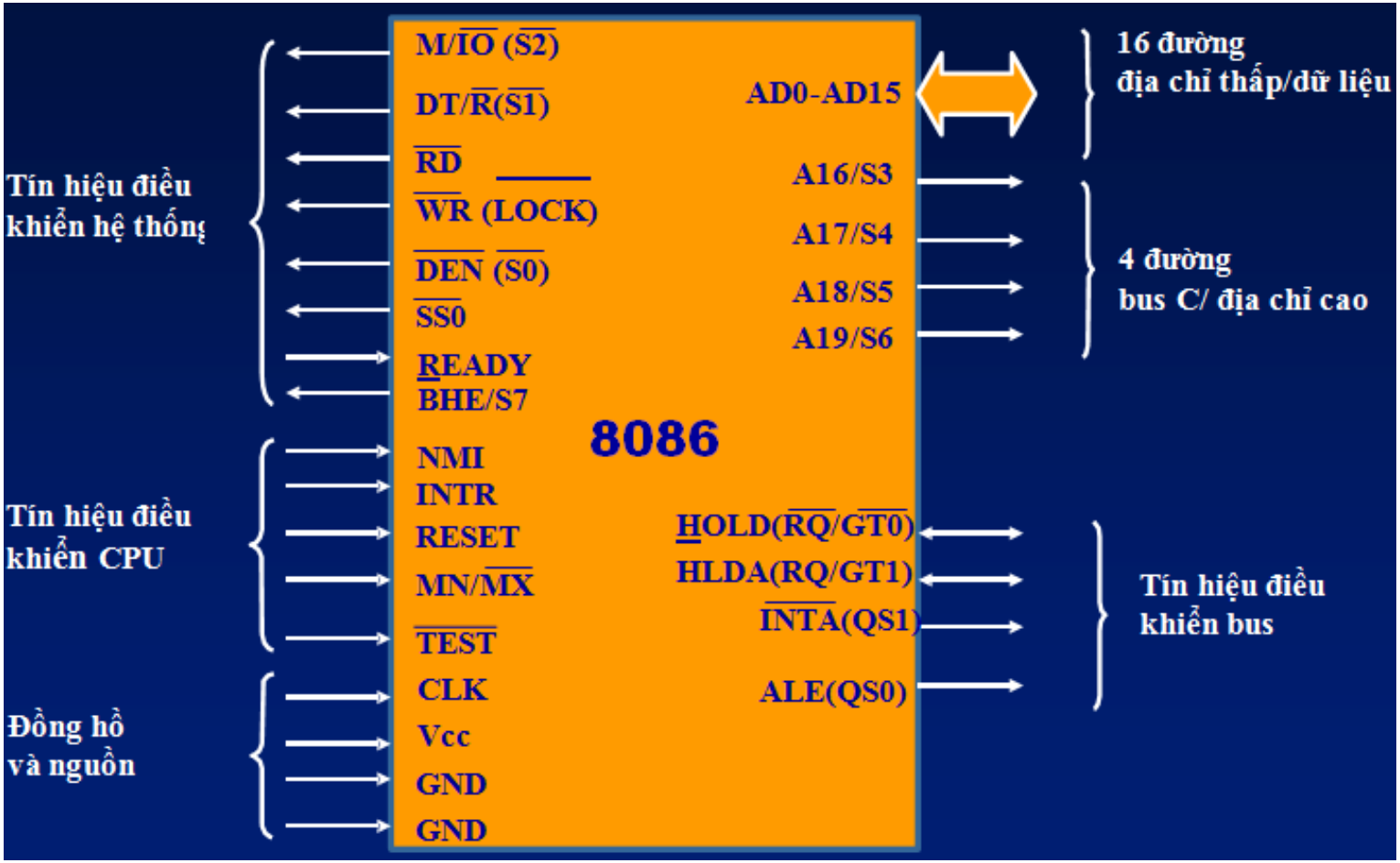
- Nhập số hệ 10
  - Số hệ 10: 0,..9 ➡ kiểm tra tính đúng đắn của dữ liệu (so sánh với 30h…39h)
  - Chuyển sang giá trị thực
  - Lưu trữ: nhân số cũ (BX) với 10 ➡ cộng tích lũy số mới vào số cũ
  - Lưu ý khi sử dụng thanh ghi AX (nhập liệu, nhân)
- Xuất số hệ 10
  - Chia liên tiếp BX cho 10 ➡ thương số = 0
  - Lưu số dư vào Stack, đếm số số dư
  - Lấy số dư trong Stack ➡ hiển thị

Ghép nối 8086 với bộ nhớ và TBNV

- Tổng quan
- Ghép nối 8086 với bộ nhớ
- Ghép nối 8086 với thiết bị ngoại vi

Tổng quan

- Các chân tín hiệu của 8086



Tổng quan

- Phân kênh và đệm bus
  - Các bus địa chỉ và dữ liệu được dùng chung
  - Nâng cao hiệu năng sử dụng bus
- Các vi mạch phân kênh và đệm bus
  - 74LS373: phân kênh
  - 74LS245: đệm dữ liệu 2 chiều
  - 74LS244: đệm 3 trạng thái theo 1 chiều

Ghép nối 8086 với bộ nhớ

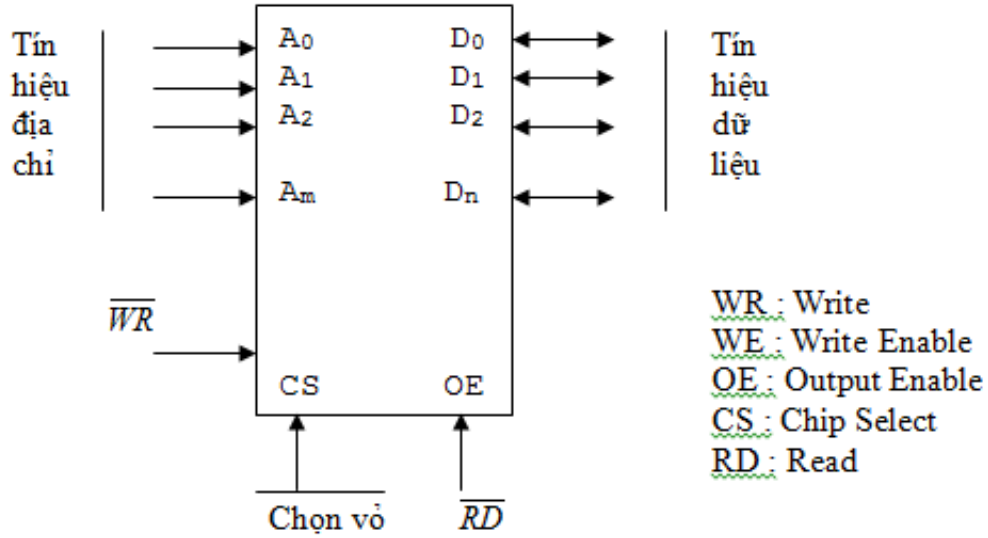
- Các loại bộ nhớ bán dẫn (ROM)
  - ROM (Read Only Memory)
  - PROM (Programmable ROM)
  - EPROM (Electrically Programmable ROM)
  - Flash
  - EEPROM (Electrically Erasable Programmable ROM)

Ghép nối 8086 với bộ nhớ

- Các loại bộ nhớ bán dẫn (RAM)
  - SRAM (Static RAM)
  - SBSRAM (Synchronous Burst RAM)
  - DRAM (Dynamic RAM)
  - FPDRAM (Fast Page mode Dynamic RAM)
  - EDODRAM (Extended Data Out Dynamic RAM)
  - SDRAM (Synchronous Dynamic RAM)
  - DDR-SDRAM (Double Data Rate SDRAM)
  - RDRAM (Rambus Dynamic RAM)
  - FeRAM (Ferroelectric Random Access Memory)
  - MRAM (Magnetoelectronic Random Access Memory)

Ghép nối 8086 với bộ nhớ

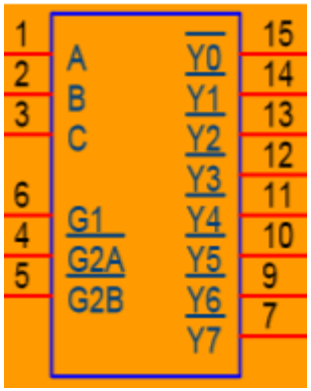
- Cấu trúc vi mạch nhớ



- Mỗi phần tử nhớ được qui chiếu tới một cách chính xác khi thực hiện các thao tác ghi/đọc.
- Được gán cho một vùng riêng biệt có địa chỉ xác định nằm trong không gian địa chỉ tổng thể của bộ nhớ
- Việc gán địa chỉ cụ thể cho mạch nhớ được thực hiện nhờ một xung chọn vỏ lấy từ mạch giải mã địa chỉ.

Ghép nối 8086 với bộ nhớ

- Giải mã địa chỉ: gán địa chỉ cụ thể cho mạch nhớ
- Các mạch giải mã:
  - Dùng cổng NAND 
  - Dùng bộ giải mã 74LS138, 74LS139



  - Dùng PROM

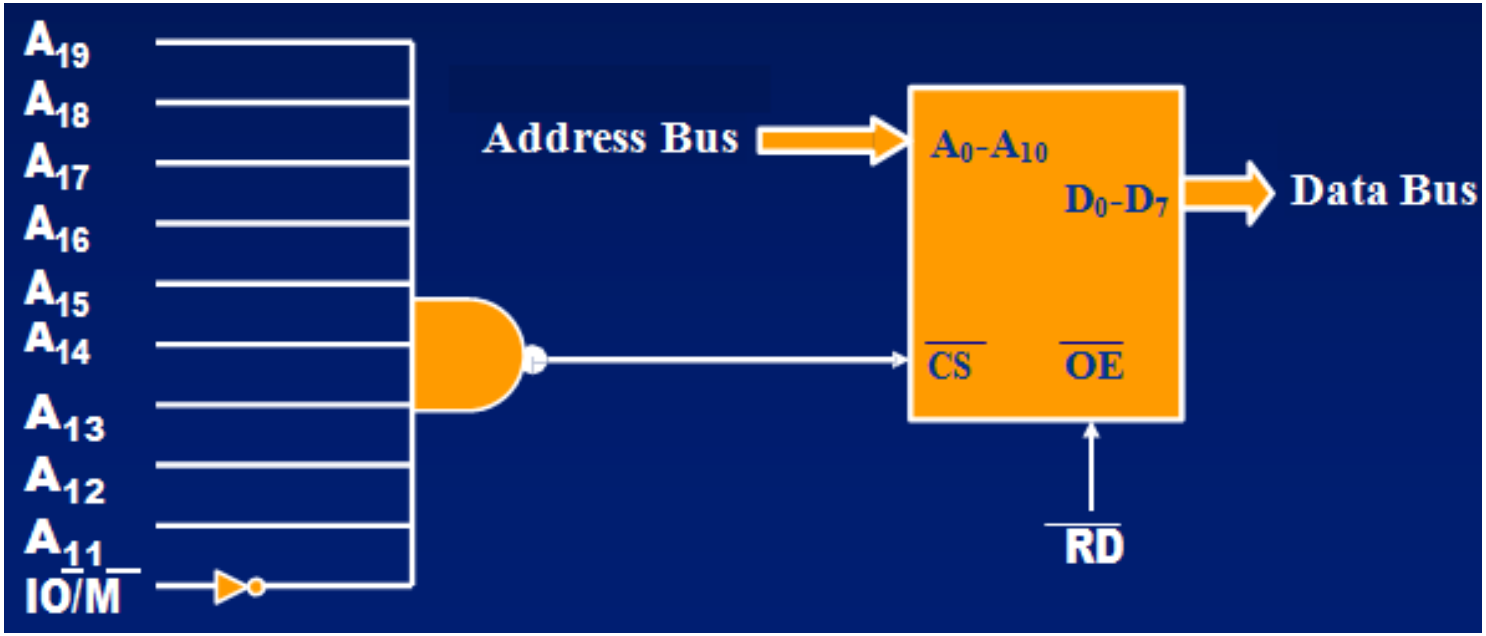
Ghép nối 8086 với bộ nhớ

- Ví dụ 1: ghép nối EPROM 2716 (2K * 8) với 8088
  - Dung lượng của EP: 2Kb (2^{11}) ➡ sử dụng 11 đường địa chỉ (A_{10} - A_0) để định địa chỉ cho EP
  - 9 đường địa chỉ còn lại của 8088 (A_{19} - A_{11}): đưa vào chân CS qua mạch NAND

	A_{19} A_{18} A_{17} A_{16}	A_{15} A_{14} A_{13} A_{12}	A_{11}	A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0
FF800:	1 1 1 1	1 1 1 1	1	0 0 0	0 0 0 0	0 0 0 0
FFFFF:	1 1 1 1	1 1 1 1	1	1 1 1	1 1 1 1	1 1 1 1

Ghép nối 8086 với bộ nhớ

- Ví dụ 1: ghép nối EPROM 2716 (2K * 8) với 8088



Ghép nối 8086 với bộ nhớ

- Ví dụ 2: giải mã địa chỉ cho vùng nhớ 64Kb, địa chỉ đầu F0000h từ các EPROM 2764 (8K*8)
  - Dung lượng của EP: 8Kb (2^{13}) ➡ sử dụng 13 đường địa chỉ (A_{12} - A_0) để định địa chỉ cho EP
  - Vùng nhớ có dung lượng 64KB cần 8EP, để giải mã cho 8 EP sử dụng bộ giải mã LS74138 ➡ các đường địa chỉ A_{15}, A_{14}, A_{13} được đưa vào các đầu vào A, B và C của LS
  - 4 đường địa chỉ còn lại của 8088 (A_{19} - A_{16}): đưa vào các cửa G_{2A}, G_{2B}, G_1 của LS.

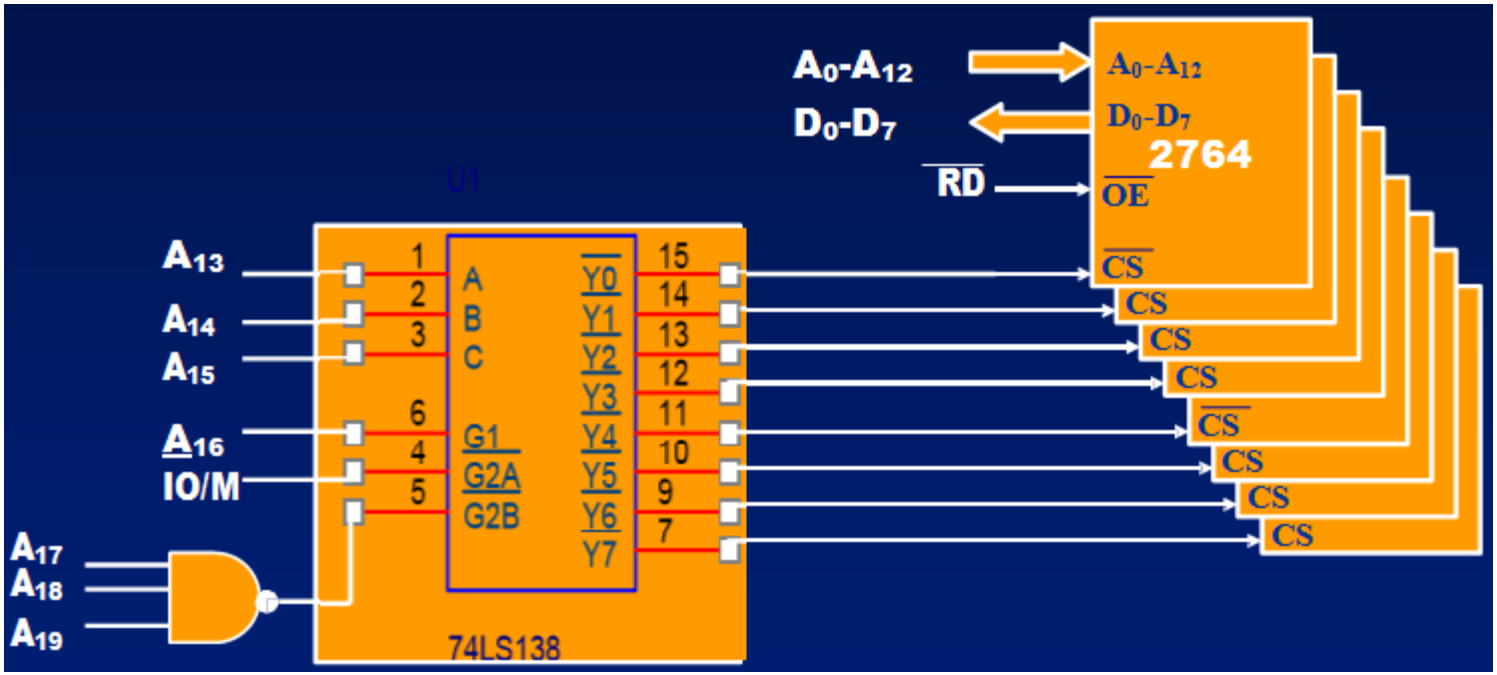
Ghép nối 8086 với bộ nhớ

- Ví dụ 2: giải mã địa chỉ cho vùng nhớ 64Kb, địa chỉ đầu F0000h từ các EPROM 2764 (8K*8)

	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}$	A_{12}	$A_{11}A_{10}A_9A_8$	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
F0000:	1 1 1 1	0 0 0	0	0 0 0 0	0 0 0 0	0 0 0 0	IC 1
F1FFF:	1 1 1 1	0 0 0	1	1 1 1 1	1 1 1 1	1 1 1 1	
F2000:	1 1 1 1	0 0 1	0	0 0 0 0	0 0 0 0	0 0 0 0	IC 2
F3FFF:	1 1 1 1	0 0 1	1	1 1 1 1	1 1 1 1	1 1 1 1	
F4000:	1 1 1 1	0 1 0	0	0 0 0 0	0 0 0 0	0 0 0 0	IC 3
F5FFF:	1 1 1 1	0 1 0	1	1 1 1 1	1 1 1 1	1 1 1 1	
...							
...							
FE000:	1 1 1 1	1 1 1	0	0 0 0 0	0 0 0 0	0 0 0 0	IC 8
FFFFF:	1 1 1 1	1 1 1	1	1 1 1 1	1 1 1 1	1 1 1 1	

Ghép nối 8086 với bộ nhớ

- Ví dụ 2: giải mã địa chỉ cho vùng nhớ 64Kb, địa chỉ đầu F0000h từ các EPROM 2764 (8K*8)



Ghép nối 8086 với bộ nhớ

- Ví dụ 3: giải mã địa chỉ cho vùng nhớ 4Kb, địa chỉ từ FF000h-FFFFFh dùng các EPROM 2716 (2K*8).

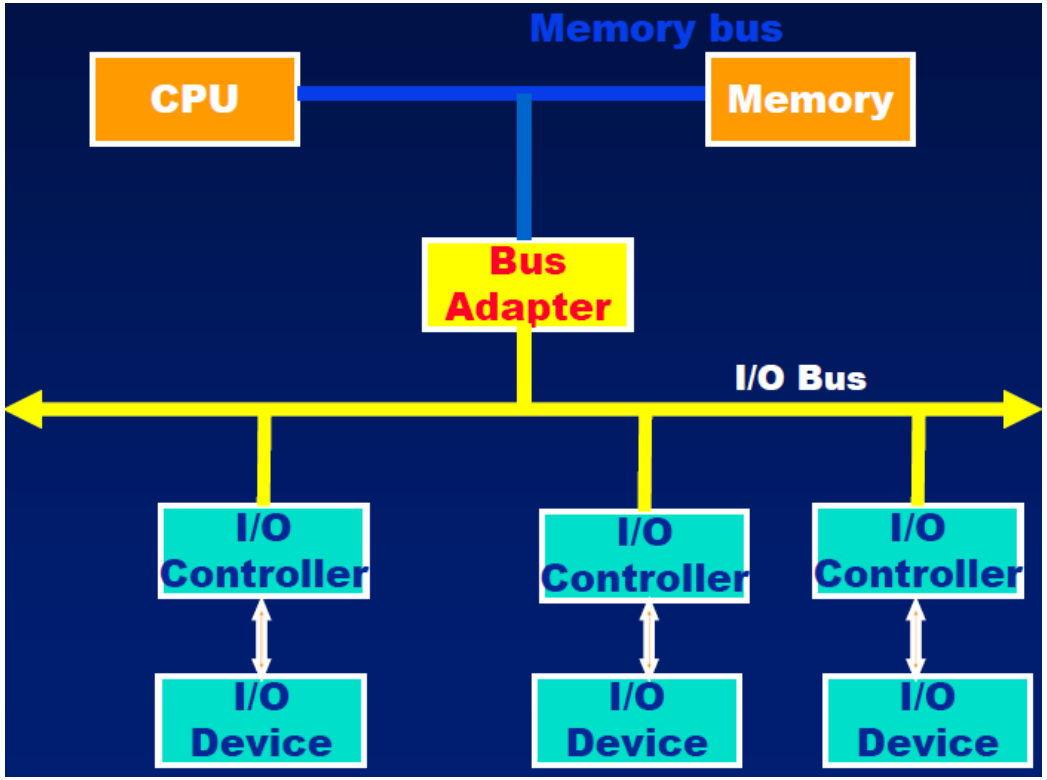
	$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}A_{12}$	$A_{11}A_{10}A_9A_8$	A_7 A_6 A_5 A_4	A_3 A_2 A_1	A_0
FF000:	1 1 1 1	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0	0
FFFFE:	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1	0
FF001:	1 1 1 1	1 1 1 1	0 0 0 0	0 0 0 0	0 0 0	1
FFFFF:	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1	1

  - Các đường A_{11}- A_1 định địa chỉ cho các EP
  - A_{19} - A_{12}, A_0: đưa vào mạch NAND

Ghép nối 8086 với bộ nhớ

- Các ví dụ khác:
  - Ghép nối 8088 với EPROM 2732 (4K*8) - 450ns.
  - Giải mã địa chỉ cho vùng nhớ 256Kb, địa chỉ đầu 00000h từ các SRAM 62256 (32K*8).
  - Giải mã địa chỉ cho vùng nhớ 128Kb, địa chỉ đầu 00000h từ các TMS DRAM 4464 (64K*4).
  - Thiết kế hệ thống nhớ cho 8086 với 64Kb EPROM và 128Kb SRAM sử dụng EPROM 27128 (16K*8) và SRAM 62256 (32K*8).

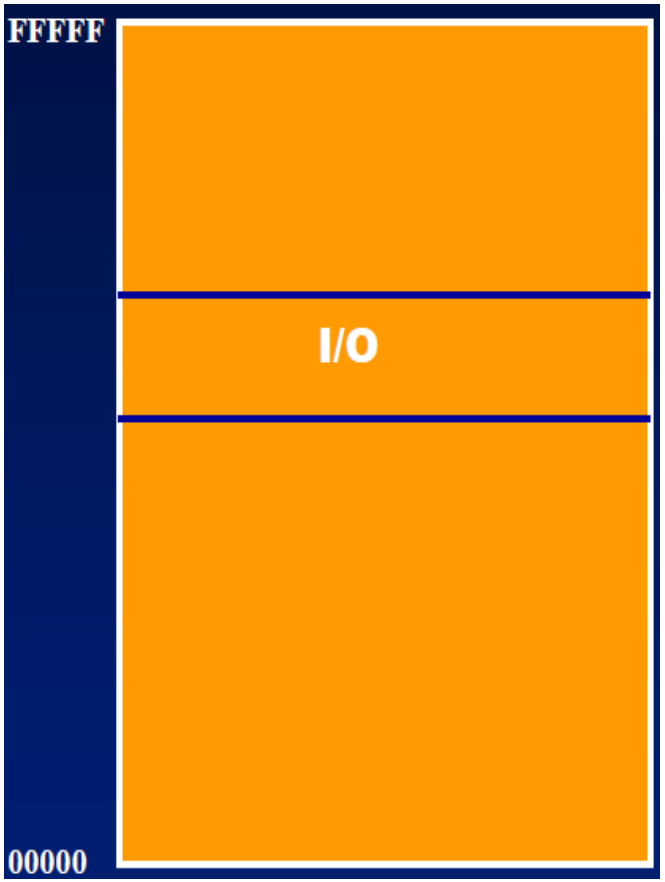
Ghép nối 8086 với TBNV



Ghép nối giữa CPU (VXL) với TBNV

Ghép nối 8086 với TBNV

Không gian địa chỉ tách biệt



Không gian địa chỉ tổng thể

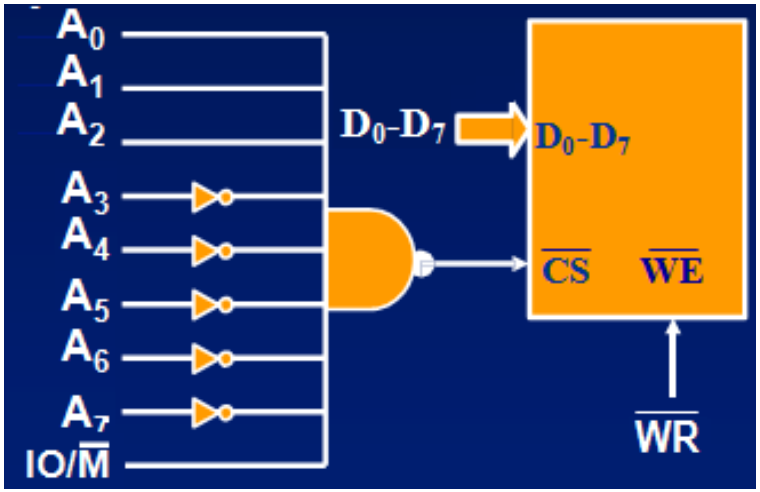
Ghép nối 8086 với TBNV

- Giải mã địa chỉ cho các cổng:
  - Các TBNV được ghép nối với VXL qua các cổng IO ➡ địa chỉ cổng
  - Số lượng TBNV < 256 ➡ sử dụng cổng 8 bit, ngược lại sử dụng cổng 16 bit
  - Đường dữ liệu: 8, 16,…bit

Ghép nối 8086 với TBNV

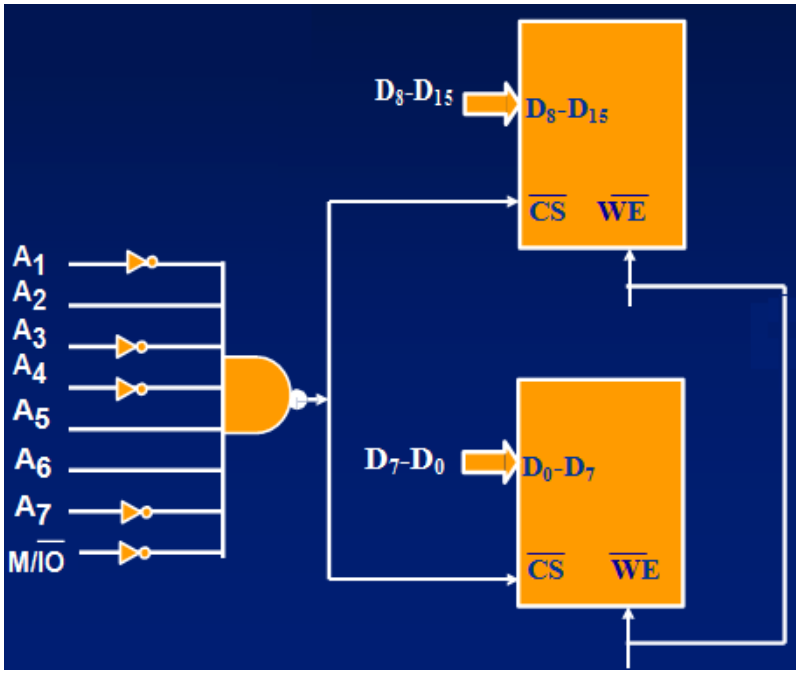
- Ví dụ 1: Giải mã địa chỉ cho thiết bị ra 8 bit với địa chỉ 07h

  07h: 0000 0111➡sử dụng các đường địa chỉ A_7 - A_0



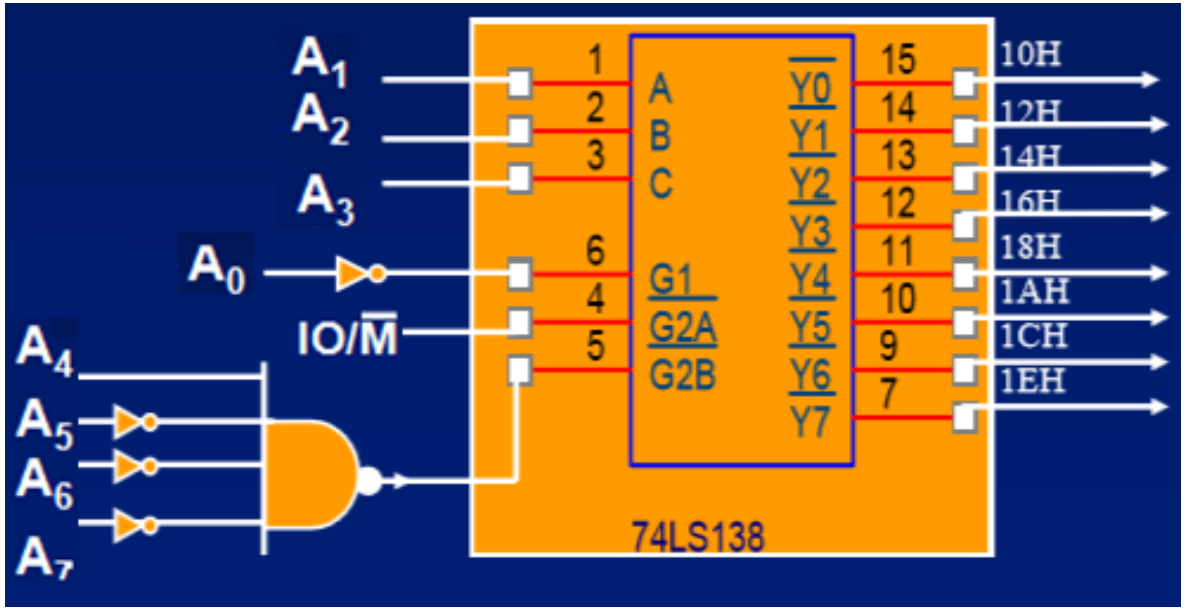
Ghép nối 8086 với TBNV

- Ví dụ 1: Giải mã địa chỉ cho thiết bị ra 16 bit với địa chỉ cổng 64h và 65h
  - 64h: 0110 0100➡sử dụng các đường địa chỉ A_7 - A_0
  - 65h: 0110 0101



Ghép nối 8086 với TBNV

- Ví dụ 1: Giải mã địa chỉ cho các cổng vào ra 8 bit ở bank thấp với các địa chỉ 10H, 12H, 14H, 16H, 18H, 1AH, 1CH, 1EH

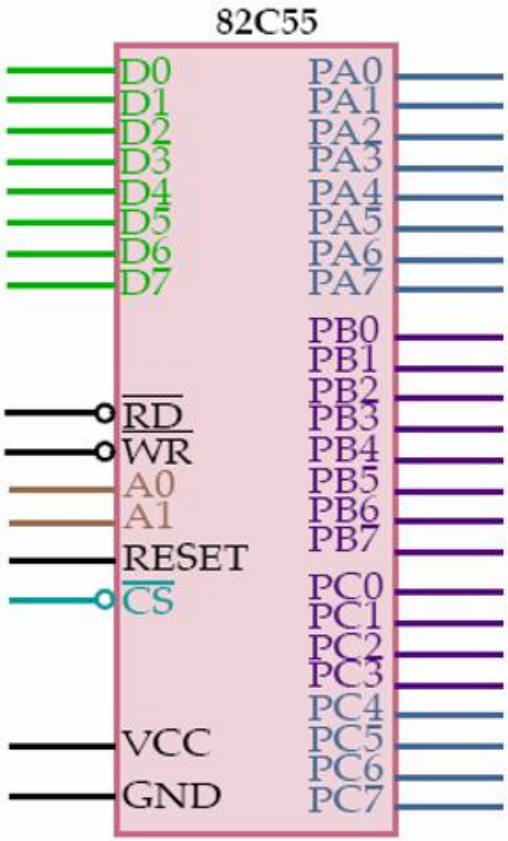


Mạch ghép nối vào/ra lập trình được

- Mạch ghép nối vào ra song song 8255A
- Mạch điều khiển 8279
- Bộ định thời 8254
- Giao tiếp truyền thông 16550
- Bộ chuyển đổi ADC – DAC
- …

Mạch ghép nối vào/ra lập trình được

- Mạch ghép nối vào ra song song 8255A



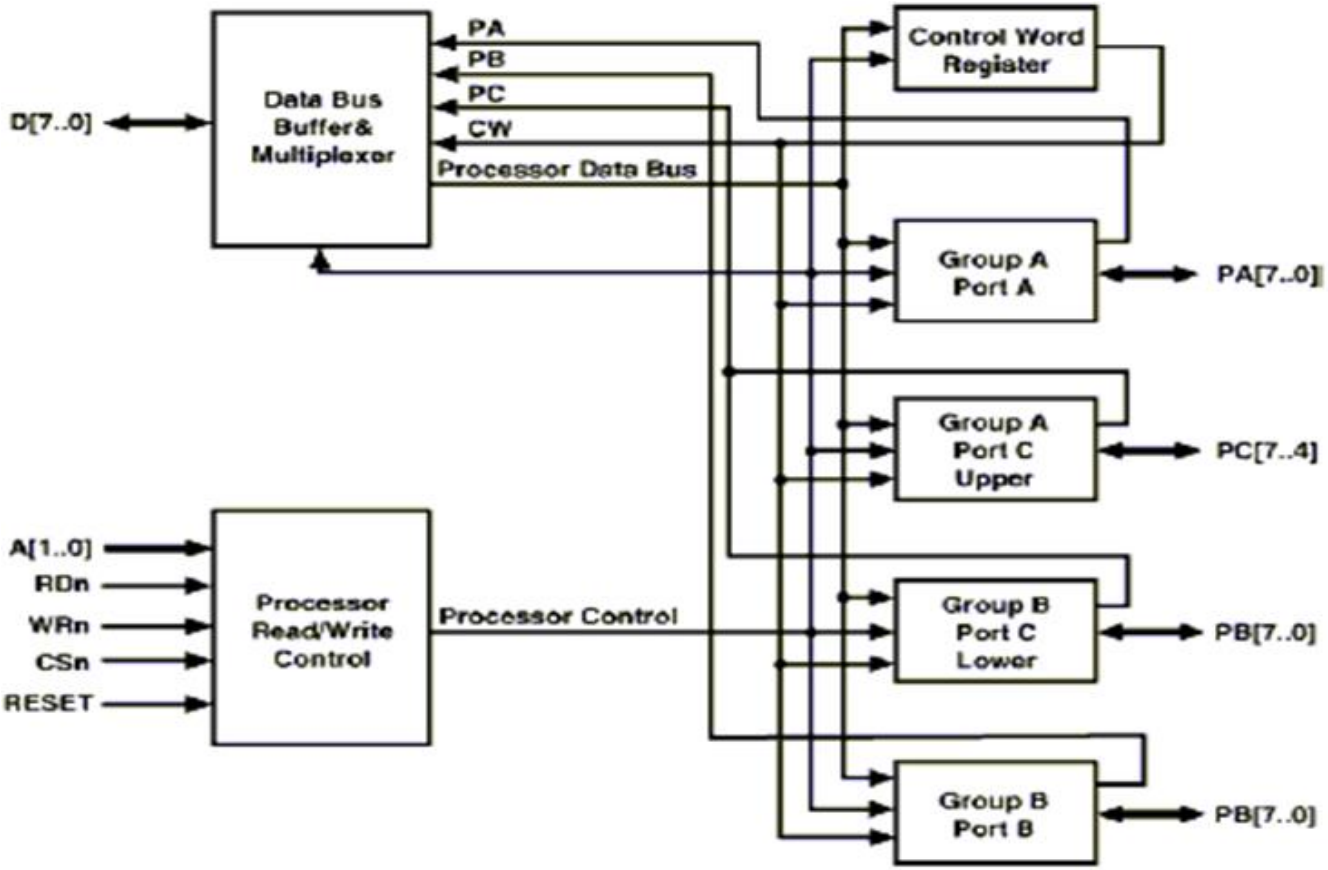
Các chân tín hiệu của 8255

A_1	A_0	*Function*
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Command Register

Thanh ghi trạng thái và các cổng

Mạch ghép nối vào/ra lập trình được

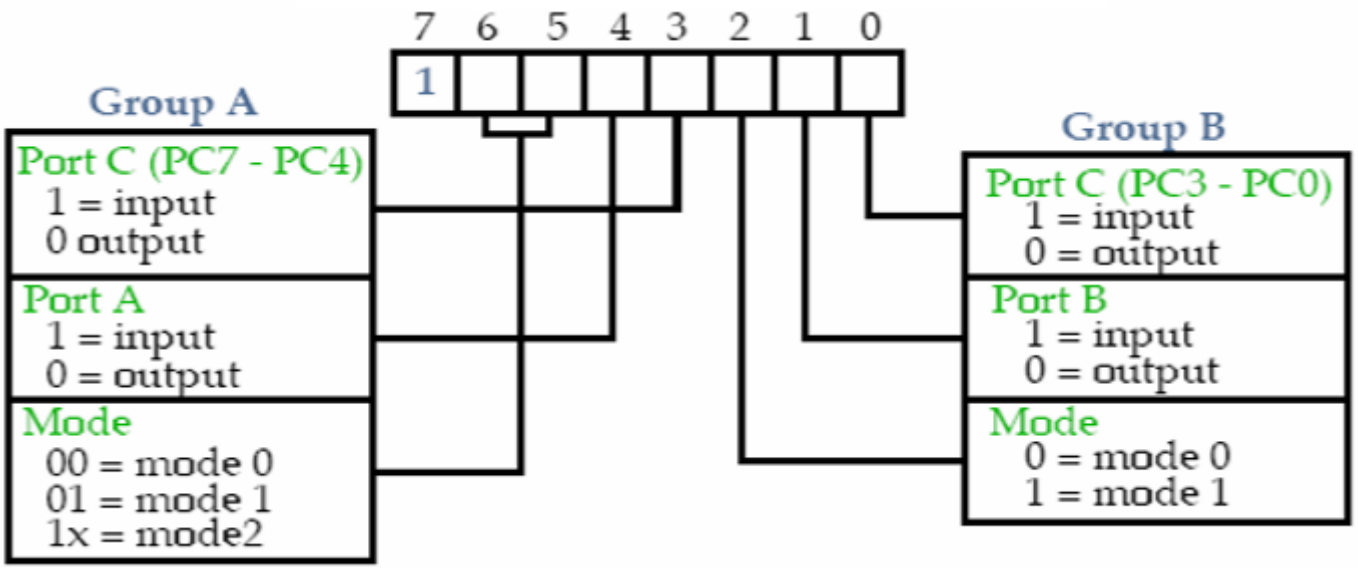
- Mạch ghép nối vào ra song song 8255A



Cấu trúc trong của 8255

Mạch ghép nối vào/ra lập trình được

- Mạch ghép nối vào ra song song 8255A



Thanh ghi từ điều khiển CWR

Mạch ghép nối vào/ra lập trình được

- Lập trình cho 8255A
  - Xác lập chế độ làm việc (mode) cho 8255
  - Xác định địa chỉ các cổng
  - Xác định từ điều khiển khởi động cấu hình hệ thống (CW)
  - Lập trình ASM
- Ví dụ: Lập trình cho 8255 hoạt động ở chế độ vào ra cơ sở giả sử địa chỉ cổng A là 7Ch. Đọc dữ liệu từ PB đưa ra PA, PC_L đưa ra PC_H

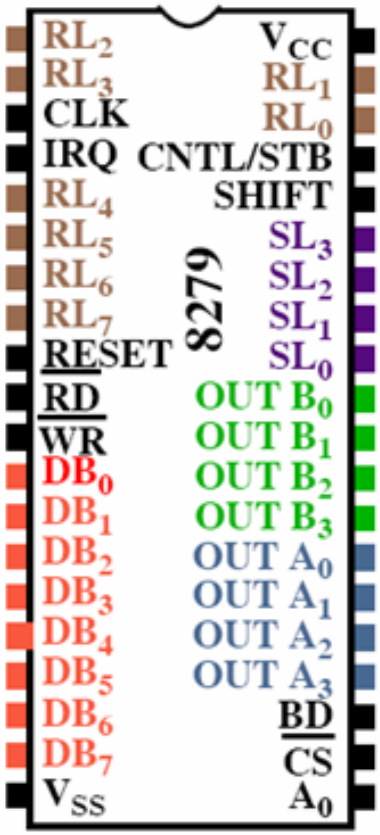
Mạch ghép nối vào/ra lập trình được

- Ví dụ
  - Chế độ làm việc của 8255: Mode 0
  - Địa chỉ các cổng lần lượt: 7Ch, 7Dh, 7Eh và 7Fh
  - Từ điều khiển: CW = 83h
  - Lập trình ASM

```
MOV AL,83h      ; từ điều khiển trong AL
OUT 7Fh,AL      ; đưa CW vào CWR
IN AL,7Dh       ; đọc cổng PB
OUT 7Ch,AL      ; đưa dữ liệu đọc được ra cổng PA
IN AL,7Eh       ; đọc cổng PCL
MOV CL,4        ; số lần quay AL
ROL AL,CL       ; chuyển 4 bit thấp thành 4 bit cao
OUT 7Eh,AL      ; đưa dữ liệu đọc được ra cổng PCH
```

Mạch ghép nối vào/ra lập trình được

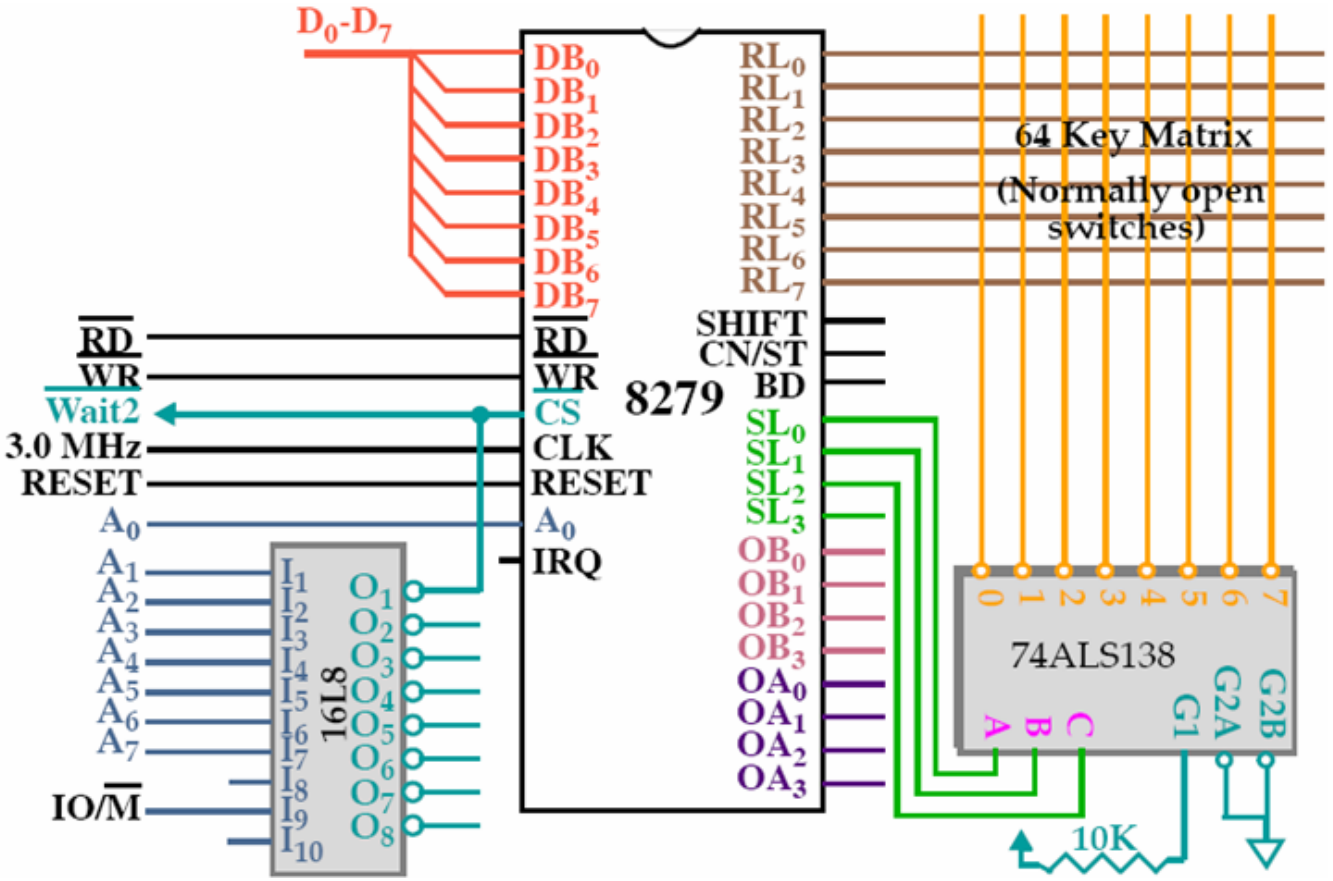
- Mạch điều khiển 8279: ghép nối bàn phím và màn hình hiển thị



Các chân tín hiệu cúa 8279

Mạch ghép nối vào/ra lập trình được

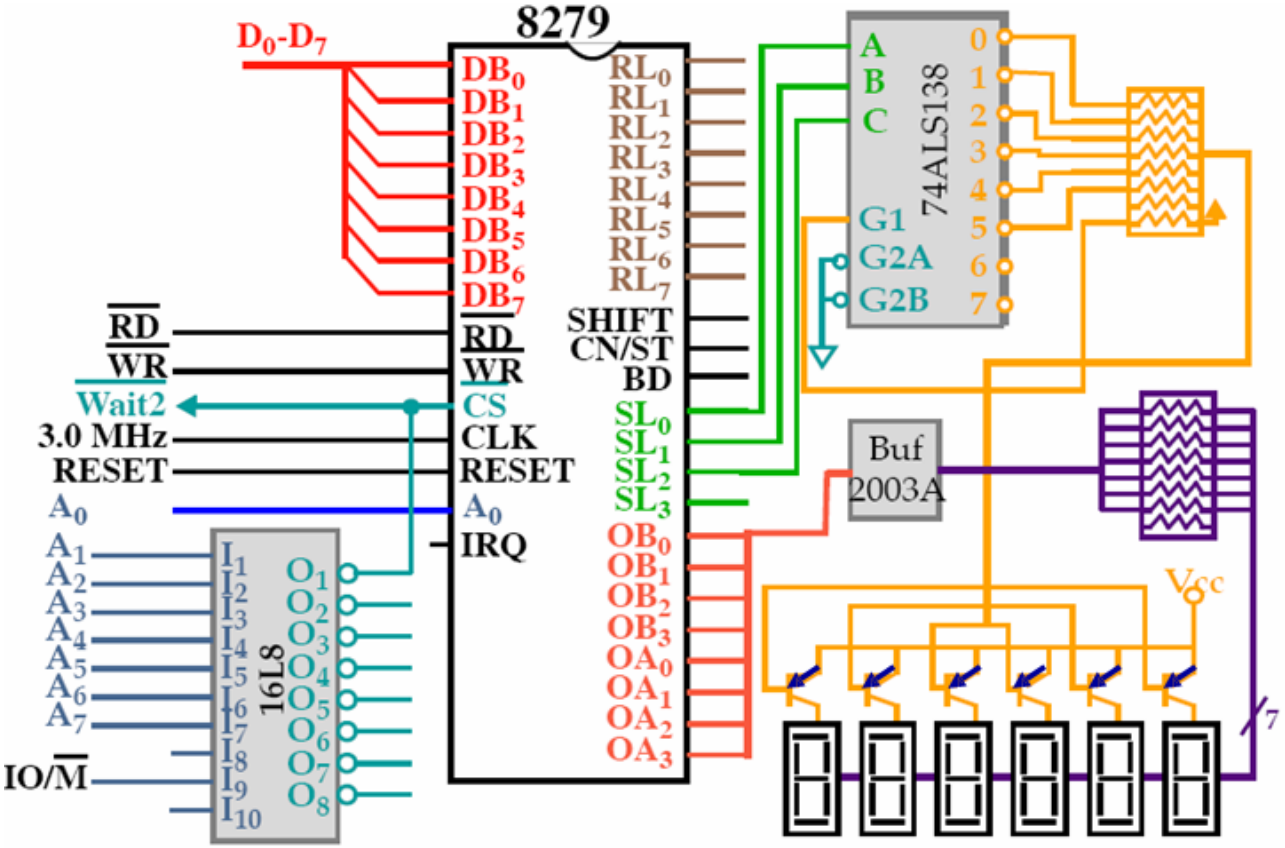
- Mạch điều khiển 8279



Ghép nối 8279 với bàn phím

Mạch ghép nối vào/ra lập trình được

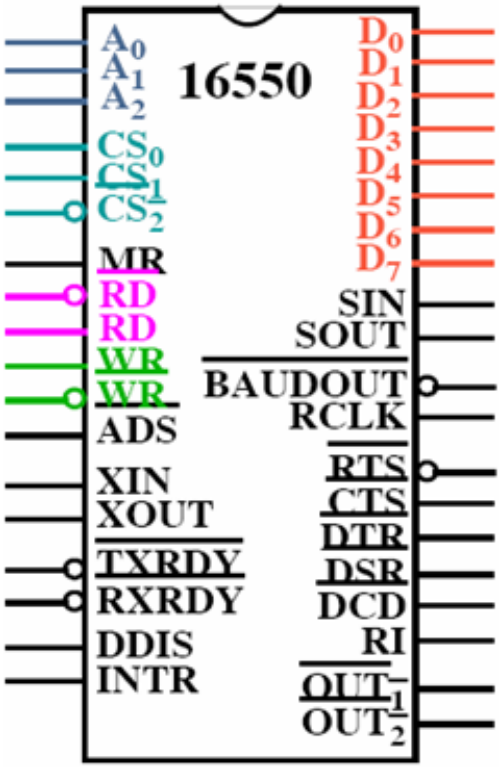
- Mạch điều khiển 8279



Ghép nối 8279 với LED 7 thanh

Mạch ghép nối vào/ra lập trình được

- Giao tiếp truyền thông 16550



Các chân tín hiệu của 16550

Tổ chức vào ra dữ liệu

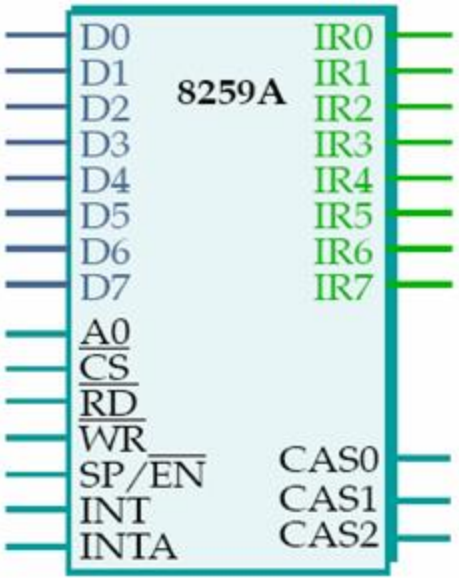
- Tổng quan
- Vào ra bằng ngắt
- Vào ra bằng truy nhập trực tiếp bộ nhớ

Tổ chức vào ra dữ liệu

- Vào/ra bằng phương pháp thăm dò, móc nối (handshaking)
  - CPU kiểm tra trạng thái của thiết bị ngoại vi
  - Nếu thiết bị ngoại vi sẵn sàng ➡ trao đổ dữ liệu
  - Nếu thiết bị ngoại vi chưa sẵn sàng ➡ CPU chờ (sau 1 khoảng thời gian nhất định sẽ phát lại tín hiệu thăm dò)
- Vào/ra bằng ngắt (Interrupt)
- Vào/ra bằng truy cập trực tiếp bộ nhớ(DMA)

Tổ chức vào ra dữ liệu

- Vào/ra bằng ngắt (Interrupt)
  - Ngắt: CPU dừng chương trình đang thực thi, đáp ứng yêu cầu của TBNV
  - Đáp ứng yêu cầu ngắt: các thao tác
  - Phân loại ngắt: cứng, mềm, ngắt khác
  - Xử lí ưu tiên ngắt: PIC (8259A)



Tổ chức vào ra dữ liệu

- Vào/ra bằng truy nhập trực tiếp bộ nhớ (Direct Memory Access - DMA)
  - Trao đổi giữa TBNV và bộ nhớ không thông qua sự điều khiển của CPU mà qua thiết bị ngoài DMAC
  - Sơ đồ, thao tác
  - Mạch điều khiển: 8237