

LẬP TRÌNH HỆ THỐNG

TS. Đỗ Quốc Huy
huydq@soict.hust.edu.vn

Các khái niệm cơ bản

9/4/2017

2

Chương 1: Các kiến thức cơ bản

Nội dung chính

- Biểu diễn dữ liệu
- Chương trình
- Kiến trúc máy tính
- Processor x86 và máy tính PC
- Phần mềm hệ thống

9/4/2017

3

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Biểu diễn dữ liệu

1. Số nhị phân
2. Phép toán với số nhị phân
3. Số hexa
4. Số có dấu
5. Các phép toán bit
6. Lưu trữ ký tự

9/4/2017 4

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Số nhị phân

- Dữ liệu được biểu diễn thông qua các giá trị 0 và 1
 - Dễ dàng để xây dựng các mạch số.
 - Ví dụ 0: 0Volts; 1 là 5Volts (3.3Volts)



- Truyền dẫn tin cậy khi gặp nhiễu
- Các giá trị 0, 1 được gọi là **bit** (binary digit)
- Các bit được nhóm lại thành byte (8 bit)
 - Byte là khối nhỏ nhất được xử lý bởi CPU

9/4/2017 5

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Biểu diễn số

- Giá trị số được biểu diễn qua dãy nhị phân
 - 1 = true; 0 = false
- bit** trong biểu diễn nhị phân được đánh số

MSB																LSB	
	1	0	1	1	0	0	1	0	1	0	0	1	1	1	0	0	
	15															0	

- MSB** –most significant bit
- LSB** –least significant bit

9/4/2017 6

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Số nguyên nhị phân không dấu

Mỗi bit mang giá trị lũy thừa 2

Mỗi số là tổng của các giá trị lũy thừa 2

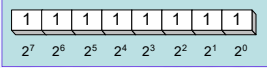


Table 1-3 Binary Bit Position Values.

2^n	Decimal Value	2^n	Decimal Value
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32768

9/4/2017 7

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Chuyển đổi Binary → Decimal

Biểu diễn A_b : $a_{n-1} \dots a_1 a_0$
 a_i là các số nhị phân (0,1), xác định giá trị

$$A = a_{n-1}2^{n-1} + \dots + a_12^1 + a_02^0 = \sum_{i=0}^{n-1} a_i2^i$$

Ví dụ: Số nhị phân A: 01101001_2 có giá trị

$$\begin{aligned} A &= 2^6 + 2^5 + 2^3 + 2^0 \\ &= 64 + 32 + 8 + 1 \\ &= 105_{(10)} \end{aligned}$$

9/4/2017 8

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Chuyển đổi Unsigned Decimal → Binary

- Chia liên tiếp cho 2
- Số dư viết theo chiều ngược lại

Ví dụ:

12 chia 2 = 6 dư 0
 6 chia 2 = 3 dư 0
 3 chia 2 = 1 dư 1
 1 chia 2 = 0 dư 1

1 1 0 0

9/4/2017 9

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Cộng trừ các số nhị phân

Phép cộng

$$\begin{array}{r} 101 \\ + 111 \\ \hline 1100 \end{array}$$

Phép trừ

$$\begin{array}{r} 1100 \\ - 111 \\ \hline 0101 \end{array}$$

9/4/2017 10

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Tổ chức dữ liệu

Máy tính làm việc với dãy bit độ dài xác định

BYTE

7 6 5 4 3 2 1 0
H.O. Nibble L.O. Nibble

WORD

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
H.O. Byte L.O. Byte

DOUBLE WORD (32bit: 0-2³²-1)

31 23 15 7 0
H.O. Word L.O. Word

QUADWORD: 64bit (0 – 2⁶⁴-1)

9/4/2017 11

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Hexadecimal integers

- Số nhị phân dài → Khó đọc → Dùng số hệ 16
- Gồm 16 ký số:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Các chữ cái A, B, C, D, E, F biểu diễn các giá trị số tương ứng trong hệ 10: 10, 11, 12, 13, 14, 15

Hệ đếm cơ số b:					
2	10	16	2	10	16
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

9/4/2017 12

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Chuyển đổi Binary ↔ Hexadecimal

- Một số Hexadecimal ứng với 4 số binary
- Nhóm 4 số binary thành 1 số hexadecimal
 - Thêm số 0 ở đầu nếu cần thiết
- Ví dụ**

$$\begin{aligned}
 &+ 52A_h = 0101\ 0010\ 1010_b = 101\ 0010\ 1010_b \\
 &+ 10111110101_b \\
 &= 0101\ 1111\ 0101_b = 5F5_h
 \end{aligned}$$

9/4/2017 13

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Chuyển đổi Hexadecimal → Decimal

- Một biểu diễn $A_h : a_{n-1} \dots a_1 a_0$
 - a_i là các số hệ hexadecimal xác định giá trị:

$$A = a_{n-1}16^{n-1} + \dots + a_116^1 + a_016^0 = \sum_{i=0}^{n-1} a_i 16^i$$

Ví dụ: Số hexa A: 2FD có giá trị

$$\begin{aligned}
 A &= 2 \times 16^2 + 15 \times 16^1 + 13 \times 16^0 \\
 &= 675
 \end{aligned}$$

9/4/2017 14

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Chuyển đổi Decimal → Hexadecimal

- Chia liên tiếp cho 16
- Viết số dư theo chiều ngược lại
- Ví dụ: $422_d = 1A6_h$

Division	Quotient	Remainder
422 / 16	26	6
26 / 16	1	A
1 / 16	0	1

9/4/2017 15

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Cộng trừ các số hệ 16

36	28	28	6A
42	45	58	4B
78	6D	80	B5

21 / 16 = 1, dư 5

16 + 5 = 21

C6	75
A2	47
24	2E

• Biến A kích thước 47byte, nằm tại địa chỉ 400020h
Biến tiếp theo có địa chỉ bao nhiêu?

9/4/2017 16

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Số nguyên có dấu

- Bit cao nhất là bit dấu (0: Số dương; 1: Số âm)
 - Số Hexa. : Chữ số cao nhất > 7, mang giá trị âm)
- Nguyên tắc biểu diễn: Mã bù 2

sign bit

1	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

Negative

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

Positive

9/4/2017 17

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Số bù 2

- Ví dụ: A là số nhị phân n bit; A' là đảo bit của A
 - $A + A' + 1 = 111...111 + 1 = 1\ 000...000 (n+1 \text{ bit}) = 0$
 - Vậy $A' + 1$ chính là -A
- Nhận xét:** Số nguyên có dấu n bit -A được biểu diễn bởi số bù 2 của A
- Nguyên tắc tính : Đảo bit và Cộng 1

Ví dụ: số nguyên có dấu 8 bit: A = -70

Biểu diễn 70 = 0100 0110

Bù 1 của 70: 1011 1001

+ 1

Bù 2 của 70: 1011 1010

Vậy: **A = 10111010**

9/4/2017 18

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Phép trừ các số nhị phân

- Nguyên tắc: Cộng với số bù 2
 - $A - B = A + (-B)$
- Ví dụ

0 1 0 1 0	0 1 0 1 0
- 0 1 0 1 1	1 0 1 0 1
	1 1 1 1 1

Nhận xét

- Dùng mã bù 2, hủy bỏ việc cần thiết có mạch logic riêng để tính phép cộng và trừ

9/4/2017 19

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Mở rộng phạm vi biểu diễn

Chuyển đổi từ số n bit sang m bit

- $n > m$
 - Số không dấu: Thêm các bit 0
 - Số có dấu: Sao chép bit dấu vào các bit mở rộng
- $n < m$
 - Chỉ thực hiện khi các bit hủy bỏ hoặc cùng là 0, hoặc cùng là 1 (số có dấu)
- Ví dụ (byte \leftrightarrow word)

9A \rightarrow FF9A	28 \rightarrow 0028
FF80 \rightarrow 80	FE40 \rightarrow lỗi

9/4/2017 20

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Phép toán logic với bit

		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

	NOT
0	1
1	0

9/4/2017 21

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Logic với số nhị phân & Chuỗi bit

- Thực hiện với từng cặp bit của 2 số

		NOT
A	1010 1010	01010101
B	0000 1111	11110000
AND	00001010	
OR	10101111	
XOR	10100101	

- Phép **AND** dùng để xoá một số bit và giữ nguyên các bit còn lại (Ví dụ: **AND** 11011011)
- Phép **OR** dùng để thiết lập 1 số bit và giữ nguyên các bit khác. (Ví dụ: **OR** 00100100)

9/4/2017 22

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Dịch bit (Shift) và phép quay (Rotate)

- Dịch trái
 - Bit thấp nhất sẽ mang giá trị 0
 - Bit cao nhất trở thành bit nhớ (carry out)
 - Tương đương với nhân giá trị với cơ số
- Dịch phải
 - Bit cao nhất sẽ mang giá trị 0
 - Bit thấp nhất trở thành bit nhớ (carry out)
- Dịch phải số học
 - Giữ nguyên bit cao nhất
 - Tương đương với chia cho cơ số
- Quay phải, quay trái

9/4/2017 23

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Trường kiểu bit và dữ liệu được đóng gói

- Một số kiểu chỉ cần sử dụng vài bit
 - Biểu diễn, ngày (5bit) tháng (4bit), giờ (5bit),...
- Đóng gói cái trường kiểu bit theo đơn vị byte
 - Hiệu quả về không gian lưu trữ
 - Xử lý chậm hơn
- Ví dụ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	M	M	M	D	D	D	D	D	Y	Y	Y	Y	Y	Y	Y

9/4/2017 24

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Ký tự và chuỗi ký tự

- Tập các ký tự
 - Bảng mã ASCII chuẩn/mở rộng
 - Bảng mã Unicode
- Chuỗi ký tự
 - Mảng các ký tự liên tiếp
 - Kết thúc bởi ký tự đặc biệt(null, ký tự \$,...)
- Các hệ thống khác nhau có thể xử lý các ký tự điều khiển theo cách khác nhau
 - Ký tự xuống dòng: CR+LF; CR; LF

9/4/2017 25

Chương 1: Các kiến thức cơ bản
1.1 Biểu diễn dữ liệu

Bài tập

1. Viết hàm chuyển số nguyên thành biểu diễn nhị phân/thập lục phân tương ứng
char * ToBinary(int) và **char * toHex(int)**
2. Viết hàm đặt, hoặc xóa một bit của số nguyên
 1. unsigned SetBit(unsigned BitMap, unsigned pos)
 2. unsigned ClrBit(unsigned BitMap, unsigned pos)
3. Viết hàm int CntBit(int) đếm số bit một của tham số
4. Viết hàm int ROR(int) và int ROL(int) thực hiện quay phải/ quay trái một bit của tham số
5. Viết hàm unsigned ToDate(int, int, int) nhận tham số ngày, tháng, năm trả về số nguyên đã mã hóa.
6. Viết 3 hàm trích ra ngày, tháng, năm tương ứng

9/4/2017 26

Chương 1: Các kiến thức cơ bản

Nội dung chính

- Biểu diễn dữ liệu
- Chương trình
- Kiến trúc máy tính
- Processor x86 và máy tính PC
- Phần mềm hệ thống

9/4/2017 27

Chương 1: Các kiến thức cơ bản
1.1 Cấu trúc chương trình

Bộ nhớ máy tính

Ví dụ: Viết chương trình hiển thị địa chỉ và nội dung các ô nhớ của một biến

9/4/2017 31

Chương 1: Các kiến thức cơ bản
1.1 Cấu trúc chương trình

Bộ nhớ máy tính

Representing Integers

Decimal: 15213
Binary: 0011 1011 0110 1101
Hex: 3 B 6 D

int A = 15213;

IA32, x86-64 Sun

6D 3B 00 00 00 00 6D

long long C = 15213;

IA32 x86-64 Sun

6D 3B 00 00 00 00 00 00 00 00 00 00 6D

int B = -15213;

IA32, x86-64 Sun

93 C4 FF FF C4 93

Two's complement representation (Covered later)

9/4/2017 32

Chương 1: Các kiến thức cơ bản
1.1 Cấu trúc chương trình

Chương trình

- Là một file có định dạng đặc biệt,
 - Chứa các thông tin cần thiết để hệ điều hành có thể đưa vào trong bộ nhớ và thực hiện
- Một chương trình bao gồm
 - Các lệnh máy, Dữ liệu được khởi tạo trước, hằng chuỗi
 - Danh sách thư viện dùng chung mà chương trình cần tới khi thực hiện
- Các kiểu file thực thi
 - Windows: .COM, .EXE,...
 - Linux: a.out, ELF,...

9/4/2017 33

Chương 1: Các kiến thức cơ bản

1.1 Cấu trúc chương trình

Chương trình

- Chương trình trong bộ nhớ chia thành nhiều phần, do các phần có tính chất khác nhau:
 - Đoạn mã lệnh (Text) cho phép đọc, thực thi; không cho phép ghi
 - Đoạn dữ liệu: Cho phép đọc, ghi, không cho phép thực hiện
- Các tiến trình sử dụng bộ nhớ độc lập với tiến trình khác
 - Vùng nhớ tiến trình quan sát là không gian địa chỉ của tiến trình

9/4/2017

34

Chương 1: Các kiến thức cơ bản

1.1 Cấu trúc chương trình

Chương trình

- TEXT** : Mã lệnh của chương trình
- RODATA** (Stores ReadOnly data):. Các hằng được chương trình sử dụng như hằng chuỗi, hằng nguyên(const int..)
- DATA** – Các biến toàn cục, được khởi tạo.
- BSS** - Các biến toàn cục không được khởi tạo. Thường được khởi tạo là zero.
- HEAP** – Vùng nhớ cấp phát động. Vùng nhớ được trả về khi gọi đến các hàm malloc/new. Tăng về phía trên cao
- SHARED LIBRARIES** – Còn được gọi là dynamic libraries. Các thư viện được sử dụng chung giữa các tiến trình.
- STACK** Lưu trữ các biến địa phương, địa chỉ trả về. Vùng nó được tăng về phía thấp.

9/4/2017

35

Chương 1: Các kiến thức cơ bản

1.1 Cấu trúc chương trình

Xây dựng chương trình

```

graph LR
    P[Programmer] --> E[Editor]
    E -- "hello.c" --> PP[C Preprocessor]
    PP -- "hello.i" --> CC[Compiler cc]
    CC --> O[Optimizer]
    O -- "hello.s" --> A[Assembler as]
    A -- "hello.o" --> L[Linker ld]
    L --> EF[Executable File hello]
    L --> SL[Static libraries .a files]
    L --> SHL[Shared Libraries .so files]
    SL -- "They add to the size of the executable." --> L
    SHL -- "Only definitions. It does not add to size of executable." --> L
    
```

9/4/2017

36

Chương 1: Các kiến thức cơ bản
1.1 Cấu trúc chương trình

Nạp chương trình

```

graph LR
    EF[Executable File] --> L[Loader  
(runtime linker)  
(usr/lib/ld.so.1)]
    SL[Shared libraries (.so, .dll)] --> L
    L --> EM[Executable in memory]
  
```

Loader sẽ được sử dụng để thực thi một file thực thi trong ngữ cảnh tiến trình

- Loader cung cấp bộ nhớ cho các phần của chương trình (Text, Data, bss,...)
- Nạp vào bộ nhớ như thư viện dùng chung (nếu chưa nạp)
- Sau khi hoàn tất, loader nhảy tới hàm `_start()`. Hàm này gọi tới hàm `init()` của các thư viện, sau đó gọi tới hàm `main()` để bắt đầu thực hiện chương trình

9/4/2017 37

Chương 1: Các kiến thức cơ bản
1.1 Cấu trúc chương trình

Thực hiện chương trình

```

graph LR
    subgraph Simple
        S1([START]) --> F1[Fetch next instruction]
        F1 --> E1[Execute instruction]
        E1 --> H1([HALT])
    end
    subgraph WithInterrupt
        S2([START]) --> F2[Fetch next instruction]
        F2 --> E2[Execute instruction]
        E2 -- "Interrupts disabled" --> H2([HALT])
        E2 -- "Interrupts enabled" --> I[Check for interrupt;  
initiate interrupt handler]
        I --> F2
    end
  
```

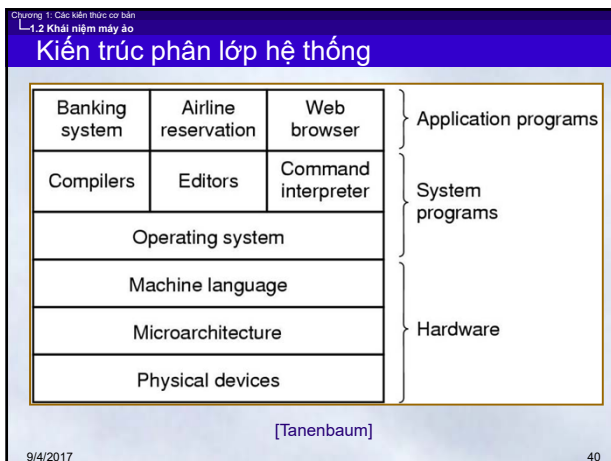
9/4/2017 38

Chương 1: Các kiến thức cơ bản

Nội dung chính

- Biểu diễn dữ liệu
- Chương trình
- Kiến trúc máy tính
- Processor x86 và máy tính PC
- Phần mềm hệ thống

9/4/2017 39

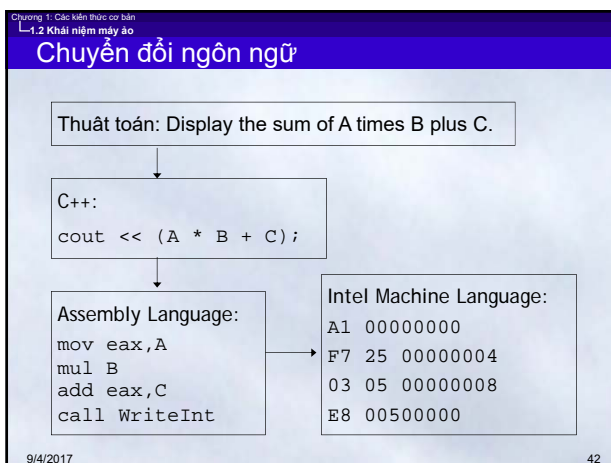


Chương 1: Các kiến thức cơ bản
1.2 Khái niệm máy ảo

Chuyển đổi ngôn ngữ

- Mỗi máy tính có một ngôn ngữ máy (ngôn ngữ L_0)
 - L_0 đơn giản, có thể được thực hiện trực tiếp bởi các mạch điện tử
 - Các ngôn ngữ « thân thiện » hơn, phải được xây dựng trên ngôn ngữ máy (ngôn ngữ L_1)
- Thực hiện chương trình bằng ngôn ngữ L_1 :
 - Translation** : Chương trình viết bằng L_1 chuyển hoàn toàn sang L_0 , sau đó thực hiện trực tiếp trên phần cứng
 - Interpretation** : Mỗi lệnh của L_1 được giải mã và thực hiện lần lượt bởi chương trình viết bằng L_0
- Có thể tồn tại nhiều lớp ngôn ngữ: L_2, L_3, \dots

9/4/2017 41



Chương 1: Các kiến thức cơ bản

1.2 Khái niệm máy ảo

Máy ảo

- Máy ảo là chương trình thực hiện chức năng của máy vật lý hoặc máy ảo khác
 - Máy ảo VM_1 thực hiện các lệnh của ngôn ngữ L_1
 - Máy ảo VM_0 thực hiện các lệnh của ngôn ngữ L_0
- Máy ảo có thể được xây dựng bởi phần cứng hoặc phần mềm
 - Xây dựng thực tế được máy VM_1 , chương trình viết bằng L_1 được thực hiện trực tiếp từ các mạch logic
 - Nếu không xây dựng được, chương trình L_1 được thực hiện bởi máy VM_0 bởi phương thức *interpreter/translation*
- Nếu L_1 chưa đủ thân thiện, tạo máy ảo VM_2 , với ngôn ngữ $L_2...$
 - Chương trình L_2 được thực hiện bởi bộ giải thích lệnh (*interpreter*) chạy trên máy VM_1 , hoặc được chuyển thành ngôn ngữ $L_1...$

Máy ảo VM_1

Máy ảo VM_0

9/4/2017

43

Chương 1: Các kiến thức cơ bản

1.2 Khái niệm máy ảo

Các mức máy riêng biệt (Tanenbaum)

High-Level Language	Level 5
Assembly Language	Level 4
Operating System	Level 3
Instruction Set Architecture	Level 2
Microarchitecture	Level 1
Digital Logic	Level 0

9/4/2017

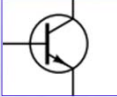
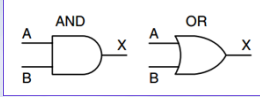
44

Chương 1: Các kiến thức cơ bản

1.2 Khái niệm máy ảo

Các mức máy ảo

- Mức 0 : Phần cứng : Bộ nhớ, Bus, CPU
 - Được tạo nên từ các mạch logic số (cổng: *gates*)
 - Các cổng logic được tạo ra từ các *Transistor*
 - Các cổng logic có một số đầu vào là bit nhị phân và tính đầu ra theo một số hàm cơ bản (AND, OR, NOT,...)
 - Các gates được kết hợp để lưu 1-bit bộ nhớ.
 - Một số 1-bit bộ nhớ kết hợp lại tạo nên thanh ghi
 - Các gates kết hợp để tạo nên khả năng tính toán

9/4/2017

45

Chương 1: Các kiến thức cơ bản
1.2 Khái niệm máy ảo

Các mức máy ảo

- Mức 1: Vi kiến trúc/Vi chương trình
 - Thực thi (interpreter) các lệnh máy ở mức 2
 - Vi chương trình (microprogram): Bộ thực thi được thực hiện bởi các mạch logic ở mức 0
 - Vi kiến trúc (micro-achitecture) Được thực hiện trực tiếp bởi phần cứng
- Mức 2: Kiến trúc tập lệnh Instruction Set Architecture
 - Gồm các lệnh cơ bản; công, trừ, AND,...
 - Các lệnh là ngôn ngữ máy
 - Các lệnh được thực hiện trực tiếp bởi
 - Phần cứng máy tính (mức 0)
 - Chương trình được nhúng trong Processor (mức 1)

9/4/2017

46

Chương 1: Các kiến thức cơ bản
1.2 Khái niệm máy ảo

Các mức máy ảo

- Mức 3: Hệ điều hành
 - Cung cấp các dịch vụ cho ứng dụng
 - Chương trình được chuyển sang mã máy để thực thi
- Mức 4: Hợp ngữ
 - Các câu lệnh dễ nhớ, được chuyển đổi 1-1 sang ngôn ngữ máy
 - Gọi các hàm được cung cấp bởi hệ điều hành
 - Chương trình được chuyển sang ngôn ngữ máy (mức 2)
- Mức 5: Ngôn ngữ bậc cao
 - Ngôn ngữ hướng ứng dụng
 - Được dịch sang hợp ngữ

9/4/2017

47

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Kiến trúc cơ bản

```

graph TD
    subgraph CPU [Central Processor Unit CPU]
        R[registers]
        ALU[ALU]
        CU[CU]
        CLK[clock]
    end
    MSU[Memory Storage Unit]
    ID1[I/O Device #1]
    ID2[I/O Device #2]
    CPU ---|data bus| MSU
    CPU ---|data bus| ID1
    CPU ---|data bus| ID2
    CPU ---|address bus| MSU
    CPU ---|address bus| ID1
    CPU ---|address bus| ID2
    CPU ---|control bus| MSU
    CPU ---|control bus| ID1
    CPU ---|control bus| ID2
    
```

- Khối xử lý trung tâm (CPU)
- Khối lưu trữ
- Thiết bị vào ra
- Đường định tuyến (bus)

9/4/2017

48

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Kiến trúc cơ bản → CPU

- Thực hiện các thao tác tính toán & logic
- Thành phần
 - Các thanh ghi:** Vùng lưu trữ trong CPU
 - Đồng hồ:** Đồng bộ hóa các thao tác
 - Khối điều khiển (C.U):** điều phối chuỗi bước thực hiện lệnh máy
 - Khối số học & logic (ALU):** thực hiện phép toán số học, logic: AND, OR,...
- Gắn với các thành phần khác qua CPU socket



9/4/2017 49

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Kiến trúc cơ bản → Khối lưu trữ

- Lưu trữ dữ liệu và lệnh của chương trình thực thi
- Thực hiện trao đổi dữ liệu từ RAM tới CPU và ngược lại
- Lưu ý:
 - Tất cả phép xử lý được thực hiện trong CPU nên phải đưa chương trình từ bộ nhớ vào CPU để thực hiện
 - Đối với CPU, thiết bị ngoại vi có thể coi là bộ nhớ vì cũng là nguồn/đích của dữ liệu

9/4/2017 50

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

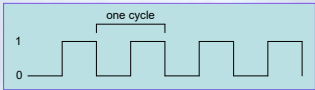
Kiến trúc cơ bản → Bus

- Các dây dẫn dùng trao đổi dữ liệu giữa các thành phần của máy tính
- Phân thành 3(4) loại
 - Data (I/O) bus:**
 - Trao đổi dữ liệu giữa CPU và bộ nhớ
 - I/O bus:** Trao đổi dữ liệu giữa CPU và thiết bị vào ra
 - Address bus:**
 - Lưu địa chỉ của câu lệnh/dữ liệu khi đang thực hiện lệnh trao đổi dữ liệu giữa CPU và bộ nhớ
 - Control bus:**
 - Sử dụng tín hiệu nhị phân để đồng bộ hoạt động của các thiết bị gắn với hệ thống bus

9/4/2017 51

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Kiến trúc cơ bản → Clock



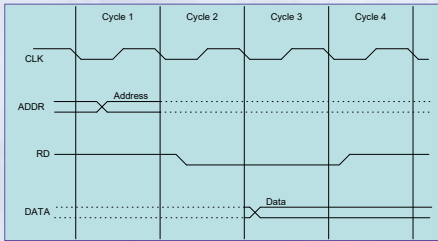
- Mục đích
 - Đồng bộ hóa các thao tác của CPU và hệ thống bus
- Chiều dài chu kỳ:
 - Thời gian hoàn thành một nhịp đồng hồ
 - 1GHz Chu kỳ 1ns
- Các lệnh máy yêu cầu từ 1 đến một vài chu kỳ
 - Truy nhập bộ nhớ cần vài chu kỳ

9/4/2017 52

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Truy nhập bộ nhớ

- Đặt bit địa chỉ của toán hạng được đặt lên Address bus (ADDR)
- Read line (RD) đặt 0 để thông báo với bộ nhớ về thao tác đọc
- CPU đợi một chu kỳ để bộ nhớ có thời gian đáp ứng. Trong chu kỳ này, khối điều khiển bộ nhớ đặt dữ liệu lên data bus (DATA)
- Read Line chuyển thành 1, báo CPU dữ liệu đã sẵn trên data bus



9/4/2017 53

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Thực hiện lệnh máy

- Gồm chuỗi các thao tác
 - Lấy lệnh (*Fetch*)
 - Giải mã (*Decode*)
 - Thực thi (*Execute*)
- Lệnh sử dụng toán hạng bộ nhớ, cần thêm
 - Lấy toán hạng (*Fetch operation*)
 - Lưu trữ kết quả (*Store output operation*)

9/4/2017 54

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Thực hiện lệnh máy

- **Fetch**
 - C.U lấy câu lệnh tiếp, tăng thanh ghi con trỏ lệnh (IP)
 - PC: Program Counter
- **Decode**
 - C.U giải mã lệnh, chỉ ra công việc cần thực hiện
 - Toán hạng cần thiết được truyền tới ALU
 - Phát tín hiệu cho ALU biết để thực thi
- **Fetch operands**
 - C.U thực hiện thao tác đọc để lấy dữ liệu
 - Lưu vào thanh ghi nội bộ
 - Thanh ghi nội bộ không truy nhập được từ chương trình

9/4/2017 55

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Thực hiện lệnh máy

- **Execute**
 - ALU thực hiện lệnh sử dụng thanh ghi chỉ ra và thanh ghi nội bộ,
 - Gửi kết quả ra thanh ghi hoặc bộ nhớ
 - ALU cập nhật trạng thái của thanh ghi cờ
- **Store output**
 - C.U sử dụng thao tác ghi để đưa kết quả ra bộ nhớ

9/4/2017 56

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Tiến trình thực hiện chương trình

Nhận được yêu cầu, hệ điều hành thực hiện

- Tìm chương trình
 - Thư mục hiện thời/trong đường dẫn hệ thống (PATH=....)
- Đọc các thông tin cần thiết của file chương trình
 - Kích thước, vị trí trên thiết bị,...
- Xin vùng nhớ còn tự do tiếp theo
 - Nạp chương trình vào vùng nhớ xin được
 - Cập nhật các thông tin về chương trình
- Thực hiện câu lệnh đầu tiên của chương trình
 - Chương trình chuyển thành tiến trình, có định danh riêng (PID)
- Tiến trình tự thực hiện
 - Hệ điều hành đáp ứng đòi hỏi tài nguyên của tiến trình
- Tiến trình kết thúc, giải phóng vùng nhớ đã cấp

9/4/2017 57

Chương 1: Các kiến thức cơ bản

Nội dung chính

- Biểu diễn dữ liệu
- Chương trình
- Kiến trúc máy tính
- Processor x86 và máy tính PC
- Phần mềm hệ thống

9/4/2017

58

Chương 1: Các kiến thức cơ bản

1.3 Kiến trúc Processor x86

Các chế độ làm việc của x86

- Real Mode
 - Chế độ cơ bản của Intel 8086.
 - Tập lệnh bị hạn chế
 - Quản lý vùng nhớ 1MByte
 - Chương trình truy nhập trực tiếp bộ nhớ, phần cứng
 - Có thể làm cho hệ thống bị lỗi
- Protected Mode
 - Chế độ làm việc chính
 - Sử dụng được tất cả các lệnh
 - Chương trình được phân các vùng nhớ riêng (đoạn)
 - Với 32-bit protected mode: Không gian địa chỉ 4GBytes
 - Ngăn các chương trình truy nhập vùng nhớ của nhau

9/4/2017

59

Chương 1: Các kiến thức cơ bản

1.3 Kiến trúc Processor x86

Các chế độ làm việc của x86

- Virtual-8086 Mode
 - Cho phép thực hiện trực tiếp các chương trình ở chế độ real mode khi processor đang protected mode
 - Mỗi chương trình thực hiện trên một máy 8086 riêng, quản lý vùng nhớ 1MByte riêng
 - Nếu chương trình lỗi, không ảnh hưởng tới hệ thống
- System Management Mode
 - Cung cấp cơ chế cài đặt các chức năng như quản lý năng lượng, an toàn hệ thống,...
 - Thường được dùng bởi nhà sản xuất để thay đổi hệ thống theo các yêu cầu đặc biệt

9/4/2017

60

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Các thanh ghi của x86

- Tên các vùng lưu trữ trực tiếp trong CPU
- Tốc độ truy nhập cao (tốc độ CPU)
- Kích thước phụ thuộc Processor: 16, 32, 64 bit
- Phân thành nhiều loại :Phổ dụng, Đoạn, Con trỏ,..

32-bit General-Purpose Registers

EAX	EBP
EBX	ESP
ECX	ESI
EDX	EDI

16-bit Segment Registers

EFLAGS	CS	ES
EIP	SS	FS
	DS	GS

9/4/2017 61

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Các thanh ghi phổ dụng

- Sử dụng cho thao tác số học, chuyển dữ liệu
- Có thể sử dụng riêng từng phần 8bit, 16 bit, 32 bit

32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

9/4/2017 62

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Các thanh ghi cơ sở và chỉ số

- Chỉ sử dụng các phần 16, 32 bit
- Source Index / Destination Index
- Base Pointer / Stack Pointer

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

9/4/2017 63

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Thanh ghi đoạn, thanh ghi cờ, con trỏ lệnh

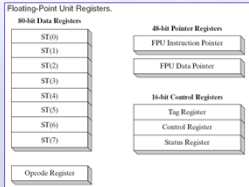
- Segment: xác định địa chỉ đoạn/bảng mô tả đoạn
 - **CS** – code segment
 - **DS** – data segment
 - **SS** – stack segment
 - **ES, FS, GS** - additional segments
- **EIP** (Extended Instruction pointer)
 - Chứa địa chỉ của lệnh được thực hiện tiếp
 - Được tăng tự động sau mỗi lệnh
 - Một số lệnh thay đổi giá trị EIP, tạo nhánh chương trình
- **EFLAGS**
 - Chứa cờ trạng thái của CPU và cờ điều khiển
 - Mỗi cờ một bit
 - Cờ được đặt nếu bằng 1 và bị xóa nếu giá trị là 0

9/4/2017 64

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Các loại thanh ghi khác

- Thanh ghi Floating-Point
 - Tồn tại trong khối xử lý số học dấu phẩy động
 - 8 thanh ghi dữ liệu 80 bit:
 - ST(0), ST(1)...ST(7)
 - Một số thanh ghi điều khiển
- Thanh ghi MMX
 - 8 thanh ghi 64 bit
 - Dùng cho các câu lệnh SIMD
 - Single-Instruction, Multiple-Data
 - Dùng trong ứng dụng đa phương tiện, truyền thông
- Thanh ghi XMM
 - 8 thanh ghi 128 bit



9/4/2017 65

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Quản lý bộ nhớ trong x86

- Real (Address) Mode
 - Segment Model
- Protected Mode (32bit)
 - Flat /Multi Segment Model
 - Paging
- Long mode
 - Processor 64bit

9/4/2017 66

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Chế độ thực

- IA32: Chế độ khi bật máy
 - x86-64bit : Trục tiếp vào chế độ bảo vệ
- Sử dụng 20 bit địa chỉ
 - Địa chỉ hóa vùng nhớ 1M: 0-FFFFFF
- Chương trình có thể truy nhập vào vị trí nhớ bất kỳ
 - Cho phép làm việc trực tiếp với phần cứng
- Đơn nhiệm
 - Cho phép ngắt tạm thời tiến trình đang thực hiện để xử lý yêu cầu từ thiết bị ngoại vi
 - Sử dụng ngắt thời gian để xây dựng chế độ đa nhiệm theo phương pháp Round Robin
 - Không có cơ chế bảo vệ → một tiến trình truy nhập vùng nhớ của tiến trình khác

9/4/2017 67

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Chế độ thực → Phương pháp phân đoạn

- Sử dụng bởi hệ điều hành MS-DOS
 - Win95, Win98 có thể khởi động vào chế độ thực
- Vấn đề:
 - Thanh ghi của 8086 16bit → Không ghi nhận địa chỉ 20bit
- Giải quyết:
 - Sử dụng bộ nhớ được phân đoạn
 - Toàn bộ bộ nhớ được chia thành các khối 64Kbytes
 - Mỗi khối gọi là một đoạn (Segment)
- Địa chỉ tuyến tính (tuyệt đối) được tính từ
 - Số hiệu đoạn 16 bit (Segment)
 - Giá trị độ lệch trong đoạn 16 bit (Offset) +

Segment + Offset = Linear address

9/4/2017 68

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Chế độ thực → Tính địa chỉ tuyến tính

$[Seg:Ofs] \rightarrow Seg \text{ SHR } 4 + Ofs$

$[08F1:0100] \rightarrow 08F1 \text{ SHR } 4 + 0100 = 08F10 + 0100 = 09010h$

linear addresses

8000:FFFF 8FFFF ← Linear Address

one segment

8000:0250

8000:0000

seg ofs

9/4/2017 69

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Chế độ thực → Nhận xét

- Các đoạn luôn bắt đầu tại vị trí chia hết 16
 - 2 đoạn liên tiếp lệch nhau 16byte (*paragraph*)
- Các đoạn chồng lên nhau
 - Một ô nhớ ứng với nhiều địa chỉ khác nhau
 - Ví dụ: địa chỉ 7C00h ứng với các địa chỉ logic
 - 0:7C00h,
 - 7B0h:100h
 - 7C0h:0

Đoạn 7B0h:100h

7C00h

7C0h:0

28F30h?

00000

FFFFF

70

9/4/2017

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Chế độ bảo vệ

- Không gian địa chỉ 4G
 - Địa chỉ hóa vùng nhớ 00000000-0FFFFFFFFh
- Mỗi tiến trình có một vùng nhớ riêng
 - Các tiến trình khác không truy nhập được
- Thiết kế cho hệ thống đa nhiệm
- Được sử dụng bởi Linux, MS-Windows
- Sử dụng các phương pháp
 - Flat Segment Model
 - Multi-Segment Model
 - Paging

9/4/2017

71

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Basic flat model

- Không gian địa chỉ liên tục, không phân đoạn
- Yêu cầu ít nhất 2 đoạn: đoạn mã lệnh và đoạn dữ liệu
- Đoạn được xác định bởi bộ mô tả đoạn (*seg. descriptor*):
 - Kích thước 64 bit, nằm trong bảng mô tả đoạn toàn cục (GDT)
- Tất cả các đoạn ánh xạ tới không gian địa chỉ vật lý 32bit :
 - Cùng địa chỉ cơ sở 0 và giới hạn 4GB
 - Không phát sinh lỗi khi sinh ra địa chỉ không có trong bộ nhớ vật lý

Figure 3-2. Flat Model

9/4/2017

72

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Protected flat model

- Cung cấp cơ chế bảo vệ phần cứng
- Tương tự basic flat mode
 - Giới hạn các đoạn được xác định
 - Lỗi sinh ra khi phát sinh địa chỉ nằm ngoài đoạn
 - Trường *access* cho biết đoạn được dùng thế nào

Figure 3-3. Protected Flat Model

9/4/2017 73

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Multi-Segment model

- Mỗi tiến trình có một bảng các mô tả đoạn riêng
 - LDT: Local Descriptor Table
 - Mỗi bộ mô tả đoạn xác định một đoạn

Local Descriptor Table

base	limit	access
00026000	0010	
00008000	000A	
00003000	0002	

RAM

- Đoạn tại vị trí 3000 có kích thước 2000h
- Đoạn tại vị trí 8000 có kích thước A000h

(Size = limit X 1000h)

9/4/2017 74

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Multi-Segment model

- Tiến trình có thể truy nhập bằng mô tả đoạn toàn cục
- Thanh ghi đoạn được gọi là Bộ chọn đoạn (*Selector*)
 - Dùng chọn phần tử trong bảng mô tả đoạn (*chỉ dùng 13bit*)

PROTECTED-MODE MEMORY MANAGEMENT

Global Descriptor Table (GDT) and Local Descriptor Table (LDT)

Segment Selector

GDT Register and LDT Register

Descriptor fields: Base address, Limit, Other fields

32-bit linear address

GDT phải chứa một (nhiều) mô tả đoạn của LDT (khi có nhiều LDT)

9/4/2017 75

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Page model

- Được hỗ trợ trực tiếp bởi CPU
- Chia mỗi đoạn thành các khối 4KB (2K, 4M) → **Page**
- Tổng bộ nhớ yêu cầu cho tất cả các chương trình có thể lớn hơn kích thước bộ nhớ vật lý
 - Một phần của chương trình nằm trong bộ nhớ
 - Một phần chương trình nằm trên đĩa → **bộ nhớ ảo**
- Khi chương trình truy nhập địa chỉ logic
 - Processor chuyển đổi đ/c logic sang địa chỉ tuyến tính
 - Sử dụng cơ chế phân trang chuyển sang địa chỉ vật lý
 - Trang chưa đưa vào bộ nhớ, sinh ra lỗi trang → **page fault**
 - Yêu cầu hệ điều hành nạp trang từ đĩa vào
 - Có thể phải ghi trang khác ra ngoài nếu cần thiết
 - Thực hiện lại câu lệnh sinh ra lỗi trang

9/4/2017 76

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Page model

- Địa chỉ logic: 16bit segment selector & 32 bit Offset
- Địa chỉ tuyến tính: 32bit xác định địa chỉ trong không gian địa chỉ tuyến tính 32bit 0 – FFFF FFFF
- Chuyển đổi địa chỉ tuyến tính sang địa chỉ vật lý, sử dụng:
 - Page Directory
 - Page Table
 - 4K-Page
- Trang 4M
 - Không Page Table

Thanh ghi CR3 32bit (12bit thấp bằng 0) xác định địa chỉ thư mục trang hiện tại

9/4/2017 77

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86 → Quản lý bộ nhớ

Intel Memory Management

Figure 3-1. Segmentation and Paging

9/4/2017 78

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Các mức vào ra

- Mức 3: Các hàm của ngôn ngữ lập trình bậc cao
 - Khả chuyển, dễ sử dụng
 - Tốc độ không phải nhanh nhất do qua nhiều lớp
- Mức 2: Hệ điều hành
 - Hệ điều hành cung cấp các dịch vụ cho ứng dụng
 - Thông qua thư viện API: Application Programming Interface
- Mức 1: BIOS
 - Được cài đặt bởi nhà sản xuất, phù hợp với phần cứng
 - Tập các dịch vụ vào ra cơ bản, mức thấp thực hiện truy nhập trực tiếp tới thiết bị
 - Một số hệ điều hành ngăn ngừa mã ứng dụng làm việc trực tiếp tại mức BIOS

9/4/2017 79

Chương 1: Các kiến thức cơ bản
1.3 Kiến trúc Processor x86

Ví dụ

```

graph TD
    A[Application Program Level 3] --> B[OS Function Level 2]
    B --> C[BIOS Function Level 1]
    C --> D[Hardware Level 0]
      
```

Hiện thị xâu ký tự

- Hàm mức 3 gọi tới hàm của hệ điều hành, truyền con trỏ xâu
- HDH sử dụng vòng lặp, thực hiện gọi dịch vụ hiện thị ký tự và dịch vụ dịch chuyển con trỏ màn hình của BIOS
- Dịch vụ của BIOS nhận ký tự, ánh xạ tới bộ font xác định và gửi tới cổng gắn với bộ điều khiển màn hình
- Bộ điều khiển màn hình, sinh ra tín hiệu để điều khiển màn hình hiển thị điểm ảnh

9/4/2017 80

Chương 1: Các kiến thức cơ bản

Các linh kiện hỗ trợ bộ xử lý trung tâm

- Giảm nhẹ công việc cả CPU
- Được dùng trong liên lạc với các tbnv
- Thay đổi/ đọc các tham số của linh kiện
 - Sử dụng câu lệnh IN, OUT
 - Làm việc với linh kiện thường phức tạp

9/4/2017 81

Chương 1: Các kiến thức cơ bản
Các linh kiện hỗ trợ bộ xử lý trung tâm

- Bộ điều khiển DMA
 - Cho phép truy nhập trực tiếp bộ nhớ
 - Đọc/Ghi dữ liệu trực tiếp từ/vào RAM không qua VXL
- Bộ điều khiển ngắt (8259)
 - Các tín hiệu ngắt đến từ các thành phần khác nhau, được đưa tới bộ điều khiển ngắt
 - Bộ điều khiển ngắt chuyển tín hiệu ngắt nào có độ ưu tiên cao hơn sẽ được đưa đến bộ xử lý trung tâm.
 - Có thể lưu giữ 15 tín hiệu ngắt
- Bộ điều khiển ghép nối các tbnv (8255)
 - Thiết lập liên lạc giữa VXL và các tbnv (bàn phím, loa,...)
 - Bộ xử lý trung tâm gọi nó để truy nhập tbnv

9/4/2017

82

Chương 1: Các kiến thức cơ bản
Các linh kiện hỗ trợ bộ xử lý trung tâm

- Bộ tạo nhịp (8254)
 - Tạo nhịp cho VXL & các linh kiện khác h/ động
 - 14.3228MHZ
 - Tần số có thể được chia để phù hợp với thiết bị
- Bộ thời gian (8253)
 - Phát ra tại các đầu ra các xung sau khoảng thời gian xác định và không đổi
 - Tần số của các xung có thể lập trình được, mỗi đầu ra có thể có một tần số nhất định
 - Các đầu ra có thể được nối với một thiết bị ngoại vi
 - Đầu ra nối với bộ điều khiển ngắt, sẽ khởi động ngắt số 8 (timer) mỗi khi nhận được một xung
 - Đầu ra nối với loa, cho phép tạo âm thanh theo tần số

9/4/2017

83

Chương 1: Các kiến thức cơ bản
Các linh kiện hỗ trợ bộ xử lý trung tâm

- Bộ điều khiển màn hình
 - Nằm trên card màn hình
 - Được cắm vào mainboard qua khe cắm mở rộng
 - Nhiệm vụ hiển thị hình ảnh trên màn hình
 - dựa vào dữ liệu trong vùng nhớ xác định trong RAM của PC
 - Có một tập các thanh ghi bên trong
 - Dùng điều khiển việc hiển thị hình ảnh lên màn hình
- Bộ điều khiển đĩa
 - Thường nằm trên các vỉ mạch mở rộng
 - Điều khiển trực tiếp hoạt động của đĩa
 - Di chuyển đầu đọc ghi đến rãnh bất kỳ, đọc/ghi DL.

9/4/2017

84

Chương 1: Các kiến thức cơ bản

Nội dung chính

- Biểu diễn dữ liệu
- Chương trình
- Kiến trúc máy tính
- Processor x86 và máy tính PC
- Phần mềm hệ thống

9/4/2017

85

Chương 1: Các kiến thức cơ bản

Phần mềm hệ thống

- Assembler
- Linker /loader
- Macro-Processor
- Operating System
 - Shell
- Compiler

9/4/2017

86

Chương 1: Các kiến thức cơ bản

Tóm tắt chương

- Biểu diễn dữ liệu
 - Số nhị phân/ Số hexa
 - Các phép toán bit
- Khái niệm máy ảo
 - Kiến trúc phân lớp hệ thống
- Kiến trúc Processor x86
 - Kiến trúc cơ bản
 - Các thanh ghi của x86
 - Quản lý bộ nhớ trong x86
 - Các mức vào ra

9/4/2017

87
