



COMPUTER ENGINEERING

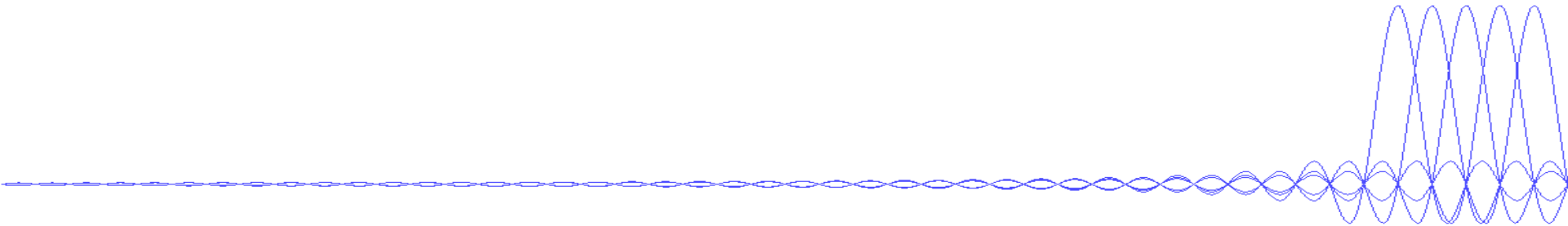
KIẾN TRÚC MÁY TÍNH



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Tuần 3

KIẾN TRÚC BỘ LỆNH





Tuần 03 – Kiến trúc bộ lệnh

Mục tiêu:

1. Hiểu cách biểu diễn và cách thực thi các lệnh trong máy tính
2. Chuyển đổi lệnh ngôn ngữ cấp cao sang assembly và mã máy
3. Chuyển đổi lệnh mã máy sang ngôn ngữ cấp cao hơn
4. Biết cách lập trình bằng ngôn ngữ assembly cho MIPS

Slide được dịch và các hình được lấy từ sách tham khảo:

Computer Organization and Design: The Hardware/Software Interface, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.



Tuần 3 – Kiến trúc bộ lệnh

- 1. Giới thiệu**
- 2. Các phép tính**
- 3. Toán hạng**
- 4. Số có dấu và không dấu**
- 5. Biểu diễn lệnh**
- 6. Các phép tính Logic**
- 7. Các lệnh điều kiện và nhảy**



Giới thiệu

❖ Để ra lệnh cho máy tính ta phải nói với máy tính bằng ngôn ngữ của máy tính. Các từ của ngôn ngữ máy tính gọi là các lệnh (*instructions*) và tập hợp tất cả các từ gọi là bộ lệnh (*instruction set*)

❖ Bộ lệnh trong chương này là MIPS, một bộ lệnh kiến trúc máy tính được thiết kế từ năm 1980. Cùng với hai bộ lệnh thông dụng nhất ngày nay:

- ARM (rất giống MIPS)
- The Intel x86



Tuần 3 – Kiến trúc bộ lệnh

- 1. Giới thiệu**
- 2. Các phép tính**
- 3. Toán hạng**
- 4. Số có dấu và không dấu**
- 5. Biểu diễn lệnh**
- 6. Các phép tính Logic**
- 7. Các lệnh điều kiện và nhảy**



Phép tính (Operations)

Ví dụ:

***add** a, b, c → Chỉ dẫn cho máy tính thực hiện cộng 2 biến b với c và ghi kết quả vào biến a,*
$$a = b + c.$$

↑
Phép tính
(operations)

↑ ↑ ↑
Toán hạng (operands)



Phép tính (Operations)

Ví dụ 1.

$a = b + c;$

$d = a - e;$



add a, b, c

sub d, a, e

C/Java

MIPS

Ví dụ 2.

$f = (g + h) - (i + j);$



add t0, g, h

add t1, i, j

sub f, t0, t1

C/Java

MIPS



Ví dụ một số lệnh trên MIPS

Category	Instruction	Example	Meaning
Arithmetic	add	add \$s1,\$s2,\$s3	$\$s1 = \$s2 + \$s3$
	subtract	sub \$s1,\$s2,\$s3	$\$s1 = \$s2 - \$s3$
	add immediate	addi \$s1,\$s2,20	$\$s1 = \$s2 + 20$
Data transfer	load word	lw \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$
	store word	sw \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1$
	load half	lh \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$
	load half unsigned	lhu \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$
	store half	sh \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1$
	load byte	lb \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$
	load byte unsigned	lbu \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$
	store byte	sb \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1$
	load linked word	ll \$s1,20(\$s2)	$\$s1 = \text{Memory}[\$s2 + 20]$
	store condition. word	sc \$s1,20(\$s2)	$\text{Memory}[\$s2 + 20] = \$s1; \$s1 = 0 \text{ or } 1$
	load upper immed.	lui \$s1,20	$\$s1 = 20 * 2^{16}$
Logical	and	and \$s1,\$s2,\$s3	$\$s1 = \$s2 \& \$s3$
	or	or \$s1,\$s2,\$s3	$\$s1 = \$s2 \$s3$
	nor	nor \$s1,\$s2,\$s3	$\$s1 = \sim (\$s2 \$s3)$
	and immediate	andi \$s1,\$s2,20	$\$s1 = \$s2 \& 20$
	or immediate	ori \$s1,\$s2,20	$\$s1 = \$s2 20$
	shift left logical	sll \$s1,\$s2,10	$\$s1 = \$s2 \ll 10$
	shift right logical	srl \$s1,\$s2,10	$\$s1 = \$s2 \gg 10$



Tuần 3 – Kiến trúc bộ lệnh

1. Giới thiệu
2. Các phép tính
- 3. Toán hạng**
4. Số có dấu và không dấu
5. Biểu diễn lệnh
6. Các phép tính Logic
7. Các lệnh điều kiện và nhảy



Có 3 loại toán hạng:

1. Toán hạng thanh ghi (Register Operands)
2. Toán hạng bộ nhớ (Memory Operands)
3. Toán hạng hằng (Constant or Immediate Operands)



Toán hạng thanh ghi:

❖ Không giống như các chương trình trong ngôn ngữ cấp cao, các toán hạng của các lệnh số học bị hạn chế, chúng phải đặt trong các vị trí đặc biệt được xây dựng trực tiếp trong phần cứng được gọi là **thanh ghi** (số lượng thanh ghi có giới hạn: MIPS-32, ARM Cortex A8-40).

❖ Kích thước của một thanh ghi trong kiến trúc MIPS là 32 bit; nhóm 32 bit xuất hiện thường xuyên nên chúng được đặt tên là “từ” (**word**) trong kiến trúc MIPS.

(Lưu ý: một “từ” trong kiến trúc bộ lệnh khác có thể không là 32 bit)

❖ Một sự khác biệt lớn giữa các biến của một ngôn ngữ lập trình và các biến thanh ghi là số thanh ghi bị giới hạn (thường là 32 thanh ghi trên các máy tính hiện nay)



Các thanh ghi trong MIPS:

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
\$zero	0	The Constant Value 0	N.A.
\$at	1	Assembler Temporary	No
\$v0-\$v1	2-3	Values for Function Results and Expression Evaluation	No
\$a0-\$a3	4-7	Arguments	No
\$t0-\$t7	8-15	Temporaries	No
\$s0-\$s7	16-23	Saved Temporaries	Yes
\$t8-\$t9	24-25	Temporaries	No
\$k0-\$k1	26-27	Reserved for OS Kernel	No
\$gp	28	Global Pointer	Yes
\$sp	29	Stack Pointer	Yes
\$fp	30	Frame Pointer	Yes
\$ra	31	Return Address	Yes



Toán hạng bộ nhớ (1):

❖ Vi xử lý chỉ có thể giữ một lượng nhỏ dữ liệu trong các thanh ghi, trong khi bộ nhớ máy tính chứa hàng triệu dữ liệu.

❖ Với lệnh MIPS, phép tính số học chỉ xảy ra trên thanh ghi, do đó, MIPS phải có các lệnh chuyển dữ liệu giữa bộ nhớ và thanh ghi. Lệnh như vậy được gọi là **lệnh chuyển dữ liệu**.

***Lệnh chuyển dữ liệu:** Một lệnh di chuyển dữ liệu giữa bộ nhớ và thanh ghi*

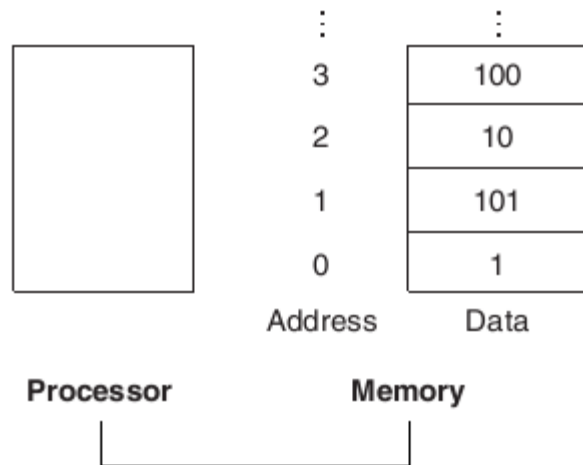
❖ Để truy cập vào một từ trong bộ nhớ, lệnh phải cung cấp **địa chỉ** bộ nhớ.

***Địa chỉ:** Một giá trị sử dụng để phân định vị trí của một phần tử dữ liệu cụ thể trong một mảng bộ nhớ.*

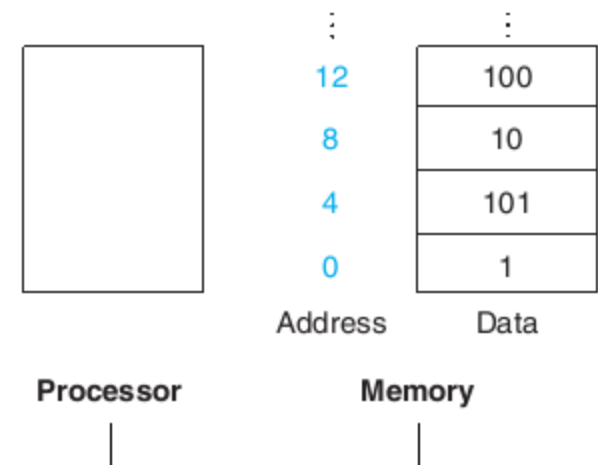


Toán hạng bộ nhớ (2):

❖ Bộ nhớ chỉ là một mảng đơn chiều lớn, với địa chỉ đóng vai trò là chỉ số trong mảng đó, bắt đầu từ 0. Ví dụ, trong hình 1, địa chỉ của phần tử thứ ba là 2, và giá trị của bộ nhớ [2] là 10.



Hình 1: Địa chỉ và nội dung của bộ nhớ giả lập như mảng.



Hình 2: Địa chỉ và nội dung bộ nhớ MIPS thực tế.

Mỗi từ nhớ (word) của MIPS là 4 bytes. MIPS định địa chỉ theo byte, địa chỉ của mỗi word là địa chỉ của byte đầu tiên trong word đó. **Do đó, địa chỉ mỗi word trong MIPS phải là bội của 4.**



Toán hạng bộ nhớ (3):

❖ Lệnh chuyển dữ liệu từ bộ nhớ vào thanh ghi gọi là nạp (load) (**viết tắt lw – load word**). Định dạng của các lệnh nạp:

lw \$s1, 20(\$s2)

Độ dời (offset)

Địa chỉ nền/cơ sở (Base address)
Thanh ghi chứa địa chỉ nền/cơ sở
gọi là thanh ghi nền/cơ sở (Base
register)

- *\$s1*: thanh ghi nạp dữ liệu vào.
- Một hằng số (20) và thanh ghi (*\$s2*) được sử dụng để truy cập vào bộ nhớ. Tổng số của hằng số và nội dung của thanh ghi này là địa chỉ bộ nhớ của phần tử cần truy cập đến. Nội dung của từ nhớ này sẽ được đưa từ bộ nhớ vào thanh ghi *\$s1*



Toán hạng bộ nhớ (4):

Ví dụ về lệnh lw:

Giả sử rằng A là một mảng của 100 phần tử (mỗi phần tử cần 1 word lưu trữ) và trình biên dịch đã kết hợp các biến g và h với các thanh ghi \$s1 và \$s2. Giả định rằng địa chỉ bắt đầu của mảng A (hay **địa chỉ cơ sở/nền**) chứa trong \$s3. Hãy biên dịch đoạn lệnh bằng ngôn ngữ C sau sang MIPS:

$$g = h + A[8];$$

→ Biên dịch:

lw \$t0, 8(\$s3) # \$t0 nhận A[8]

add \$s1,\$s2,\$t0 # g = h + A[8]

Thực tế trong MIPS một word là 4 bytes, do đó lệnh đúng phải là:

lw \$t0, 32(\$s3)

❖ Hằng số trong một lệnh truyền dữ liệu (8) gọi là **offset**, và thanh ghi chứa địa chỉ bắt đầu của mảng (\$s3) gọi là **thanh ghi cơ sở**.



Toán hạng bộ nhớ (5):

❖ Lệnh chuyển dữ liệu từ thanh ghi ra bộ nhớ, gọi là lệnh lưu (store) (**viết tắt sw – store word**). Định dạng của các lệnh lưu:

sw \$s1, 20(\$s2)

offset

Base address in base register

- \$s1: thanh ghi chứa dữ liệu cần lưu.
- Một hằng số (20) và thanh ghi (\$s2) được sử dụng để truy cập vào bộ nhớ. Tổng số của hằng số và nội dung của thanh ghi này là địa chỉ bộ nhớ, nơi mà nội dung đang chứa trong thanh ghi \$s1 sẽ được lưu vào đây.



Toán hạng bộ nhớ (6):

Ví dụ lệnh *sw*:

Giả sử biến h được kết nối với thanh ghi $\$s2$ và địa chỉ cơ sở của mảng A là trong $\$s3$. Biên dịch câu lệnh C thực hiện dưới đây sang MIPS?

$$A[12] = h + A[8];$$

$12 * 4 = 48$ $8 * 4 = 32$

→ Biên dịch:

lw $\$t0, 32(\$s3)$

$\$t0 = A[8]$

add $\$t0, \$s2, \$t0$

$\$t0 = h + A[8]$

sw $\$t0, 48(\$s3)$

$A[12] = \$t0$



Toán hạng bộ nhớ (7):

- ❖ **Alignment Restriction:** Trong MIPS, các từ phải bắt đầu từ địa chỉ là bội số của 4. Yêu cầu này được gọi là một “**alignment restriction**” và nhiều kiến trúc hiện nay buộc tuân theo quy định này nhằm giúp việc truyền dữ liệu nhanh hơn. Tuy nhiên một số kiến trúc vẫn không bắt buộc quy định này.

(Chú ý: Tại sao tuân theo điều này giúp truyền dữ liệu nhanh hơn → đọc chương 5 sách tham khảo chính)

- ❖ Leftmost - “**Big End**”, “**Big Endian**”

Rightmost - “**Little End**”, “**Little Endian**”

➔ MIPS thuộc dạng nào?



Toán hạng hằng:

Một hằng số/số tức thời (constant/immediate number) có thể được sử dụng trong một phép toán

Ví dụ:

addi \$s3, \$s3, 4

\$s3 = \$s3 + 4



Toán hạng hằng



Tóm lại, chỉ có 3 loại toán hạng trong một lệnh của MIPS

1. Toán hạng thanh ghi (Register Operands)
2. Toán hạng bộ nhớ (Memory Operands)
3. Toán hạng hằng (Constant or Immediate Operands)

Lưu ý:

- ❖ Các hằng số trong MIPS có thể âm nên không cần phép trừ một thanh ghi và một số tức thời trong MIPS.
- ❖ Trong thực tế, có một phiên bản khác của MIPS làm việc với các thanh ghi 64 bits, gọi là MIPS-64. MIPS xem xét trong môn học này là MIPS làm việc với các thanh ghi chỉ 32 bit, gọi là MIPS-32.

⇒ Trong phạm vi môn học này, MIPS dùng chung sẽ hiểu là MIPS-32



Tuần 3 – Kiến trúc bộ lệnh

1. Giới thiệu
2. Các phép tính
3. Toán hạng
4. Số có dấu và không dấu
5. Biểu diễn lệnh
6. Các phép tính Logic
7. Các lệnh điều kiện và nhảy



Số có dấu và không dấu

- ❖ Con người được dạy để suy nghĩ trong hệ cơ số 10, nhưng con số có thể được biểu diễn trong bất kỳ cơ số nào. Ví dụ, 123 cơ số 10 = 1111011 cơ số 2.
- ❖ Số lưu trữ trong máy tính như một chuỗi các tín hiệu điện thế cao và thấp, do đó chúng được xem như hệ cơ số 2.

Ví dụ: Hình vẽ dưới đây cho thấy như thế nào một word của MIPS lưu trữ số 1011:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

(32 bits wide)

- ❖ Một word của MIPS có 32 bit, do đó có thể biểu diễn các số từ 0 đến $2^{32}-1$ (4.294.967.295)
- ❖ **Bit trọng số nhỏ nhất (*The least significant bit – LSB*):** Bit ngoài cùng bên phải trong một từ nhớ (bit 0)
- ❖ **Bit trọng số lớn nhất (*The most significant bit – MSB*):** Bit ngoài cùng bên trái trong một từ nhớ (bit 31)



Số có dấu và không dấu

❖ Số dương và âm trong máy tính:

Các máy tính hiện tại sử dụng **bù hai** để biểu diễn nhị phân cho **số có dấu**.

- Nếu MSB = 0: số dương
- Nếu MSB = 1: số âm.

→ Bit thứ 32 (MSB) còn được gọi là **bit dấu**.

```
0000 0000 0000 0000 0000 0000 0000 0000two = 0ten
0000 0000 0000 0000 0000 0000 0000 0001two = 1ten
0000 0000 0000 0000 0000 0000 0000 0010two = 2ten
... ..
```

```
0111 1111 1111 1111 1111 1111 1111 1101two = 2,147,483,645ten
0111 1111 1111 1111 1111 1111 1111 1110two = 2,147,483,646ten
0111 1111 1111 1111 1111 1111 1111 1111two = 2,147,483,647ten
1000 0000 0000 0000 0000 0000 0000 0000two = -2,147,483,648ten
1000 0000 0000 0000 0000 0000 0000 0001two = -2,147,483,647ten
1000 0000 0000 0000 0000 0000 0000 0010two = -2,147,483,646ten
... ..
```

```
1111 1111 1111 1111 1111 1111 1111 1101two = -3ten
1111 1111 1111 1111 1111 1111 1111 1110two = -2ten
1111 1111 1111 1111 1111 1111 1111 1111two = -1ten
```




Số có dấu và không dấu

- ❖ Nửa phần dương của các con số, từ 0 đến $2,147,483,647_{\text{ten}}$ ($2^{31} - 1$), biểu diễn như thường.
- ❖ Phần số âm biểu diễn:

$$1000 \dots 0000_{\text{two}} = -2,147,483,648_{\text{ten}}$$

$$1000 \dots 0001_{\text{two}} = -2,147,483,647_{\text{ten}}$$

$$1111 \dots 1111_{\text{two}} = -1_{\text{ten}}$$

- ❖ Bù hai có một số âm $-2,147,483,648_{\text{ten}}$, mà không có số dương tương ứng.

$$0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_{\text{two}} = 0_{\text{ten}}$$

$$0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0001_{\text{two}} = 1_{\text{ten}}$$

$$0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0010_{\text{two}} = 2_{\text{ten}}$$

...

...

$$0111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1101_{\text{two}} = 2,147,483,645_{\text{ten}}$$

$$0111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1110_{\text{two}} = 2,147,483,646_{\text{ten}}$$

$$0111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111_{\text{two}} = 2,147,483,647_{\text{ten}}$$

$$1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000_{\text{two}} = -2,147,483,648_{\text{ten}}$$

$$1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0001_{\text{two}} = -2,147,483,647_{\text{ten}}$$

$$1000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0010_{\text{two}} = -2,147,483,646_{\text{ten}}$$

...

...

$$1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1101_{\text{two}} = -3_{\text{ten}}$$

$$1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1110_{\text{two}} = -2_{\text{ten}}$$

$$1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111_{\text{two}} = -1_{\text{ten}}$$



Số có dấu và không dấu

Công thức chuyển từ một số bù hai sang số hệ 10:

$$(x_{31} \times -2^{31}) + (x_{30} \times 2^{30}) + (x_{29} \times 2^{29}) + \dots + (x_1 \times 2^1) + (x_0 \times 2^0)$$

Lưu ý: Bit dấu được nhân với -2^{31} , và phần còn lại của các bit sau đó được nhân với các số dương của các giá trị cơ số nào tương ứng của chúng.

Ví dụ: đổi từ hệ 2 sang hệ 10

1111 1111 1111 1111 1111 1111 1111 1100_{two}

Trả lời:

$$\begin{aligned} & (1 \times -2^{31}) + (1 \times 2^{30}) + (1 \times 2^{29}) + \dots + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) \\ &= -2^{31} + 2^{30} + 2^{29} + \dots + 2^2 + 0 + 0 \\ &= -2,147,483,648_{\text{ten}} + 2,147,483,644_{\text{ten}} \\ &= -4_{\text{ten}} \end{aligned}$$



Số có dấu và không dấu

Mở rộng số có dấu:

Làm thế nào để chuyển đổi một số nhị phân được biểu diễn trong n bit thành một số biểu diễn với nhiều hơn n bit?

Ví dụ:

Chuyển đổi số nhị phân 16 bit của số 2_{ten} và -2_{ten} thành số nhị phân 32 bit.

→ 2_{ten} : 0000 0000 0000 0010_{two} → 0000 0000 0000 0000 0000 0000 0000 0010_{two}

→ -2_{ten} : 1111 1111 1111 1110_{two} → 1111 1111 1111 1111 1111 1111 1111 1110_{two}



Số có dấu và không dấu

Khi làm việc với các lệnh của MIPS, lưu ý:

- Mở rộng có dấu (Sign-extend)
- Mở rộng không dấu (Zero-extend)



Tuần 3 – Kiến trúc bộ lệnh

Tổng kết:

- Giới thiệu lệnh máy tính, tập lệnh là gì
(Tập lệnh được sử dụng cụ thể trong môn học này là MIPS 32 bits)
- Tập lệnh bao gồm các nhóm lệnh cơ bản: Nhóm lệnh logic, nhóm lệnh số học, nhóm lệnh trao đổi dữ liệu và nhóm lệnh nhảy
- Với MIPS, toán hạng cho các lệnh được chia thành ba nhóm: nhóm toán hạng thanh ghi, nhóm toán hạng bộ nhớ và nhóm toán hạng là số tức thời
- Nhắc lại số có dấu và số không dấu



Tuần 3 – Kiến trúc bộ lệnh

❖ Lý thuyết: Đọc sách tham khảo

- Mục: 2.1, 2.2, 2.3, 2.4
- Sách: *Computer Organization and Design: The Hardware/Software Interface*, Patterson, D. A., and J. L. Hennessy, Morgan Kaufman, Revised Fourth Edition, 2011.

❖ Bài tập: file đính kèm