

KIẾN TRÚC MÁY TÍNH

Dr. Nguyen Dang Khoa

Email: khoa.nguyendang@phcnika-univ.edu.vn
khoaulsan@gmail.com

Phone: 0974.844.618

TẬP LỆNH MIPS

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

1. GIỚI THIỆU

- ❖ Cần một phương thức để giao tiếp với máy tính → Sử dụng ngôn ngữ của máy tính → 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

2. CÁC PHÉP TÍNH

❖ Java, C, C++

a=b+c;
d=a-e;

❖ MIPS ???

2. CÁC PHÉP TÍNH

❖ Java, C, C++

a=b+c;
d=a-e;

❖ MIPS

Java, C, C++	MIPS
a=b+c;	add a,b,c
d=a-e;	sub d,a,e

2. CÁC PHÉP TÍNH

❖ Java, C, C++

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

❖ MIPS ???

2. CÁC PHÉP TÍNH

❖ Java, C, C++

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

❖ MIPS

Java, C, C++	MIPS
$f = (g + h) - (i + j);$	add t0, g, h add t1, i, j sub f, t0, t1

2. CÁC PHÉP TÍNH

Nhóm	Lệnh	Ý nghĩa	Nhóm	Lệnh	Ý nghĩa
Arithmetic	add	Add	Logical	and	And
	sub	Subtract		or	Or
	addi	Add immediate		nor	Nor
Data transfer	lw	Load word		andi	And immediate
	sw	Store word		ori	Or immediate
	lh	Store half		sll	Shift left logical
	lb	Load byte		srl	Shift right logical
	lbu	Load byte unsigned			
	sb	Store byte			
	ll	Load linked word			
	sc	Store condition, word			
	lui	Load upper immediate			

3. TOÁN HẠNG

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

3.1. TOÁN HẠNG THANH GHI

- ❖ Ở slide trước:
a=b+c; ➔ add a,b,c (toán hạng b,c và tổng a ở đây được coi là các biến)
- ❖ Tuy nhiên trong MIPS, không giống như các chương trình trong ngôn ngữ cấp cao, **các toán hạng của các lệnh số học bị hạn chế**, chúng phải sử dụng các **thanh ghi** được quy định sẵn.
- ❖ Kích thước của một thanh ghi trong kiến trúc MIPS là 32 bit (đặt tên **word** trong kiến trúc MIPS. (Một số kiến trúc bộ lệnh khác có thể không là 32 bit)

3.1. TOÁN HẠNG THANH GHI

Tên thanh ghi	Số thứ tự	Ý nghĩa
\$zero	0	The Constant value 0
\$at	1	Assembler Temporary
\$v0-\$v1	2-3	Values for function results and Expression Evaluation
\$a0-\$a3	4-7	Arguments
\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved Temporaries
\$t8-\$t9	24-25	Temporaries
\$k0-\$k1	26-27	Reserved for OS Kernel
\$gp	28	Global Pointer
\$sp	29	Stack Pointer
\$fp	30	Frame Pointer
\$ra	31	Return Address

3.2. TOÁN HẠNG BỘ NHỚ

- ❖ Với lệnh MIPS, phép tính số học chỉ xảy ra trên thanh ghi, do đó, phải sử dụng lưu trữ trong các bộ nhớ để làm trung gian. 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.
- ❖ 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)

- \$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
- \$s1 = MEM[\$s2 + 20]

3.2. TOÁN HẠNG BỘ NHỚ

❖ Ví dụ.

$$g = h + A[4];$$

#Độc được giá trị trong A[4]. Ví dụ \$s3 lưu địa chỉ của A[0]
lw \$t0, 16(\$s3) # \$t0 nhận A[4] # \$t0=MEM(\$s3+16)=A[4]
add \$s1,\$s2,\$t0 # g = h + A[4] # \$s1=\$s2+\$t0=h+A[4]

❖ Tại sao là 16

3.2. TOÁN HẠNG BỘ NHỚ

❖ Ví dụ.

$$g = h + A[4];$$

lw \$t0, 16(\$s3) # \$t0 nhận A[4]
add \$s1,\$s2,\$t0 # g = h + A[4]

❖ Tại sao là 16: Thực tế trong MIPS một word là 4 bytes, do đó lệnh đúng phải là:
lw \$t0, 16(\$s3)

3.2. TOÁN HẠNG BỘ NHỚ

❖ 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)

- \$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 thanh ghi trong \$s1 sẽ được lưu trữ trong bộ nhớ này.

lw \$s1,20(\$s2) ⇔ \$s1 = MEM[\$s2+20]
sw \$s1,20(\$s2) ⇔ MEM[\$s2+20]=\$s1

3.2. TOÁN HẠNG BỘ NHỚ

❖ Ví dụ: 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?

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

3.2. TOÁN HẠNG BỘ NHỚ

❖ Ví dụ: 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?

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

lw \$t0,32(\$s3) # \$t0 = A[8]
add \$t0,\$s2,\$t0 # \$t0 = h + A[8]
sw \$t0,48(\$s3) # A[12] = \$t0

A0				A1				A2				A3				A4			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16				

3.3. TOÁN HẠNG HẲNG

❖ 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

4. BIỂU DIỄN LỆNH

- Máy tính chỉ có thể làm việc với các tín hiệu điện tử thấp và cao, do đó một lệnh lưu giữ trong máy tính phải được biểu diễn như là một chuỗi của "0" và "1", được gọi là mã máy/lệnh máy.
- Ngôn ngữ máy (Machine language): biểu diễn nhị phân được sử dụng để giao tiếp trong một hệ thống máy tính.
- Để chuyển đổi từ một lệnh sang mã máy (machine code) sử dụng định dạng lệnh (instruction format).

Định dạng lệnh: Một hình thức biểu diễn của một lệnh bao gồm các trường của số nhị phân.

Ví dụ một định dạng lệnh:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

4. BIỂU DIỄN LỆNH

- Để chuyển đổi từ một lệnh sang mã máy (machine code) sử dụng định dạng lệnh (instruction format).

Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address

- ✓ "reg" nghĩa là chỉ số thanh ghi (giữa 0 và 31)
- ✓ "address" nghĩa là 1 địa chỉ 16 bit.
- ✓ "n.a." (không áp dụng) nghĩa là trường này không xuất hiện trong định dạng này

4. BIỂU DIỄN LỆNH

- Ví dụ: Chuyển đổi một lệnh cộng trong MIPS thành một lệnh máy: **add \$t0,\$s1,\$s2** Với định dạng lệnh:

Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address

\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved Temporaries

op	rs	rt	rd	shamt	funct
000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

4. BIỂU DIỄN LỆNH

Tên thanh ghi	Số thứ tự	Ý nghĩa
\$zero	0	The Constant value 0
\$at	1	Assembler Temporary
\$v0-\$v1	2-3	Values for funtion results and Expression Evaluation
\$a0-\$a3	4-7	Arguments
\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved Temporaries
\$t8-\$t9	24-25	Temporaries
\$k0-\$k1	26-27	Reserved for OS Kernel
\$gp	28	Global Pointer
\$sp	29	Stack Pointer
\$fp	30	Frame Pointer
\$ra	31	Return Address

Bài tập 1: (2p)
sub \$t3,\$s3,\$s0

add \$t0,\$s1,\$s2

op	Rs (\$s1)	Rt (\$s2)	Rd (\$t0)	shamt	funct
000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

4. BIỂU DIỄN LỆNH

Tên thanh ghi	Số thứ tự	Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
\$zero	0									
\$at	1	add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
\$v0-\$v1	2-3	sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
\$a0-\$a3	4-7	addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
\$t0-\$t7	8-15	lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
\$s0-\$s7	16-23	sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
\$t8-\$t9	24-25									
\$k0-\$k1	26-27									
\$gp	28									
\$sp	29									
\$fp	30									
\$ra	31									

Bài tập 1: (2p)
sub \$t3,\$s3,\$s0

op	Rs (\$s3)	Rt (\$s0)	Rd (\$t3)	shamt	funct
000000	10011	10000	01011	00000	100010
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

4. BIỂU DIỄN LỆNH

Tên thanh ghi	Số thứ tự	Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
\$zero	0									
\$at	1	add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
\$v0-\$v1	2-3	sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
\$a0-\$a3	4-7	addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
\$t0-\$t7	8-15	lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
\$s0-\$s7	16-23	sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
\$t8-\$t9	24-25									
\$k0-\$k1	26-27									
\$gp	28									
\$sp	29									
\$fp	30									
\$ra	31									

Bài tập 2: (2p)
add \$t3,\$s3,\$s0

op	Rs (\$s3)	Rt (\$t3)	Address
001000	10011	01011	0000 0000 0001 1110
6 bits	5 bits	5 bits	5 bits

4. BIỂU DIỄN LỆNH

Tên thanh ghi	Số thứ tự	Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
\$zero	0									
\$at	1	add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
\$v0-\$v1	2-3	sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
\$a0-\$a3	4-7	addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
\$t0-\$t7	8-15	lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
\$s0-\$s7	16-23	sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
\$t8-\$t9	24-25									
\$k0-\$k1	26-27									
\$gp	28									
\$sp	29									
\$fp	30									
\$ra	31									

Bài tập 3: (2p)
addi \$s2,\$s5,24

op	Rs (\$s)	Rt (\$s2)	Address
001000	10101	10010	0000 0000 0001 1000
6 bits	5 bits	5 bits	5 bits 5 bits 6 bits

4. BIỂU DIỄN LỆNH

- ❖ Bài tập 4 (4 điểm): Biên dịch câu lệnh C thực hiện dưới đây sang MIPS (1 điểm). Từ đó viết mã lệnh cho các lệnh (3 điểm) (10p)

$m = h + k + 34;$
 $m = m + 5;$

4. BIỂU DIỄN LỆNH

- ❖ Bài tập 4 (4 điểm): Biên dịch câu lệnh C thực hiện dưới đây sang MIPS (1 điểm). Từ đó viết mã lệnh cho các lệnh (3 điểm) (10p)

$m = h + k + 34;$
 $m = m + 5;$

Lời giải:

Giá sử h, k, m được lưu trữ trong thanh ghi \$t1, \$t2, \$t3. Lệnh MIPS cho đoạn chương trình C được biểu diễn như sau:

add \$t4, \$t1, \$t2 // h+k → 00,0000 0,1001 0,1010 0,1100 0,0000 10,0000
addi \$t3, \$t4, 34 //m+34 → 00,1000 0,1100 0,1011 0,0000 0,0000 10,0010
addi \$t3, \$t3, 5 //m=m+5 → 00,1000 0,1011 0,1011 0,0000 0,0000 00,0101

Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
addi	I	8 ₁₀	Reg	n.a	n.a	n.a	n.a	Constant
lw	I	35 ₁₀	Reg	n.a	n.a	n.a	n.a	Address \$t0-\$t7
sw	I	43 ₁₀	Reg	n.a	n.a	n.a	n.a	Address

8-15 Temporaries

4. BIỂU DIỄN LỆNH

❖ Summary:

Rs, Rt, Rd

Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address

4. BIỂU DIỄN LỆNH

❖ Biểu diễn lệnh lw và sw:

Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address

op	rs	rt	Constant or address
6 bits	5 bits	5 bits	16 bits

❖ Ví dụ: lw \$t0,1500(\$t1)

4. BIỂU DIỄN LỆNH

❖ Ví dụ: lw \$t0,1500(\$t1)

Lệnh	Định dạng	Op	Rs	Rt	Rd	shamt	Funct	Address
add	R	0	Reg	Reg	Reg	0	32 ₁₀	n.a
sub	R	0	Reg	Reg	Reg	0	34 ₁₀	n.a
addi	I	8 ₁₀	Reg	Reg	n.a	n.a	n.a	Constant
lw	I	35 ₁₀	Reg	Reg	n.a	n.a	n.a	Address
sw	I	43 ₁₀	Reg	Reg	n.a	n.a	n.a	Address

\$t0-\$t7	8-15	Temporaries
-----------	------	-------------

op	rs	rt	Constant or address
100011	01001	01000	0000 0101 1101 1100

4. BIỂU DIỄN LỆNH

- ❖ Chuyển câu lệnh sau sang assembly MIPS và sau đó chuyển thành mã máy:
 $A[200] = h + A[200]$
- ❖ Biết A là một mảng nguyên, mỗi phần tử của A cần một từ nhớ để lưu trữ; \$t1 chứa địa chỉ nền/cơ sở của mảng A và \$s2 tương ứng với biến nguyên h.

4. BIỂU DIỄN LỆNH

- ❖ Chuyển câu lệnh sau sang assembly MIPS và sau đó chuyển thành mã máy:
 $A[200] = h + A[200]$
- ❖ Biết A là một mảng nguyên, mỗi phần tử của A cần một từ nhớ để lưu trữ; \$t1 chứa địa chỉ nền/cơ sở của mảng A và \$s2 tương ứng với biến nguyên h.

lw \$t0,\$s2(\$t1) # Dùng thanh ghi tạm \$t0 nhận A[300]
add \$t0,\$s2,\$t0 # Dùng thanh ghi tạm \$t0 nhận $h + A[300]$
sw \$t0,\$s2(\$t1) # Lưu $h + A[300]$ trở lại vào A[300]

op	Rs	Rt	Rd	shamt	funct
100011	01001	01000		0000 0011 0010 0000	
000000	10001	01000	01000	00000	100000
101011	01001	01000		0000 00 11 0010 0000	

5. CÁC PHÉP TÍNH LOGIC

- ❖ Một số phép tính logic

- ✓ **Shift:** Lệnh dịch chuyển bit.
- ✓ **AND:** là phép toán logic "VÀ".
- ✓ **OR:** là một phép toán logic "HOẶC".
- ✓ **NOT:** kết quả là 1 nếu bit đó là 0 và ngược lại.
- ✓ **NOR:** NOT OR.

Phép tính Logic	C/C++	Java	MIPS	opcode
Shift left	<<	<<	sll	000000
Shift right	>>	>>	srl	000000
Bit-by-bit AND	&	&	andi	000000
Bit-by-bit OR			ori	000000
Bit-by-bit NOT	~	~	nor	000000

- ✓ Hằng số rất hữu ích trong các phép toán logic AND và OR cũng như trong phép tính số học, vì vậy MIPS cung cấp các lệnh trực tiếp **andi** và **ori**.

5. CÁC PHÉP TÍNH LOGIC

Phép tính Logic	MIPS	Syntax	Meaning	opcode	Rs	Rt	Rd
Shift left	sll	SLL Rd,Rt,h	Rd=Rt<<h	000000	Any	5 bits	5 bits	000000
Shift right	srl	SRL Rd,Rt,h	Rd=Rt>>h	000000	Any	5 bits	5 bits	000010
Bit-by-bit AND	and	AND Rd, Rs, \$Rt	\$Rd = \$Rs & \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0100
	andi	ANDI Rt, Rs, imm	\$Rt = \$Rs & imm	001100	5 bits	5 bits	No	16 bits
Bit-by-bit OR	or	OR Rd, Rs, Rt	\$Rd = \$Rs \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0101
	ori	ORI Rt, Rs, imm	\$Rt = \$Rs imm	001101	5 bits	5 bits	No	16 bits
Bit-by-bit NOT	nor	NOR Rd,Rs,Rt	Rd=~(\$Rt)	000000	5 bits	5 bits	5 bits	000 0010 0111

MIPS	Giải thích
sll \$t1,\$t3,2	# Assum t3=1 (0001) # \$t1 = 1<<2 = (0100) = 4
sll \$t1,\$t3,2	
opcode	Rs (\$t0) Rt (\$t3) Rd (\$t1) (11 bits) = 2 + ...
000000	0.1000 0.1011 0.1001 00010.000000

5. CÁC PHÉP TÍNH LOGIC

Phép tính Logic	MIPS	Syntax	Meaning	opcode	Rs	Rt	Rd	Bit cuối (16bits)
Shift left	sll	SLL Rd,Rt,h	Rd=Rt<<h	000000	Any	5 bits	5 bits	h.000000
Shift right	srl	SRL Rd,Rt,h	Rd=Rt>>h	000000	Any	5 bits	5 bits	h.000010
Bit-by-bit AND	and	AND Rd, Rs, \$Rt	\$Rd = \$Rs & \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0100
	andi	ANDI Rt, Rs, imm	\$Rt = \$Rs & imm	001100	5 bits	5 bits	No	16 bits
Bit-by-bit OR	or	OR Rd, Rs, Rt	\$Rd = \$Rs \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0101
	ori	ORI Rt, Rs, imm	\$Rt = \$Rs imm	001101	5 bits	5 bits	No	16 bits
Bit-by-bit NOT	nor	NOR Rd,Rs,Rt	Rd=~(\$Rt)	000000	5 bits	5 bits	5 bits	000 0010 0111

MIPS	Giải thích
srl \$t2,\$t1,5	
srl \$t2,\$t1,5	
opcode	Rs (\$t0) Rt (\$t1) Rd (\$t2) (11 bits) = 5 + ...
000000	0.1000 0.1001 0.1010 00101.000010

5. CÁC PHÉP TÍNH LOGIC

Phép tính Logic	MIPS	Syntax	Meaning	opcode	Rs	Rt	Rd	Bit cuối (16bits)
Shift left	sll	SLL Rd,Rt,h	Rd=Rt<<h	000000	Any	5 bits	5 bits	h.000000
Shift right	srl	SRL Rd,Rt,h	Rd=Rt>>h	000000	Any	5 bits	5 bits	h.000010
Bit-by-bit AND	and	AND Rd, Rs, \$Rt	\$Rd = \$Rs & \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0100
	andi	ANDI Rt, Rs, imm	\$Rt = \$Rs & imm	001100	5 bits	5 bits	No	16 bits
Bit-by-bit OR	or	OR Rd, Rs, Rt	\$Rd = \$Rs \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0101
	ori	ORI Rt, Rs, imm	\$Rt = \$Rs imm	001101	5 bits	5 bits	No	16 bits
Bit-by-bit NOT	nor	NOR Rd,Rs,Rt	Rd=~(\$Rt)	000000	5 bits	5 bits	5 bits	000 0010 0111

MIPS	Giải thích
andi \$t2,\$t1,5	
andi \$t2,\$t1,5	
opcode	Rs (\$t1) Rt (\$t2) 16 bits (5)
001100	0.1001 0.1010 0.0000.0000.0000.101

5. CÁC PHÉP TÍNH LOGIC

Phép tính Logic	MIPS	Syntax	Meaning	opcode	Rs	Rt	Rd	Bit cuối (16bits)
Shift left	sll	SLL Rd,Rt,h	Rd=Rt<<h	000000	Any	5 bits	5 bits	h.000000
Shift right	srl	SRL Rd,Rt,h	Rd=Rt>>h	000000	Any	5 bits	5 bits	h.000010
Bit-by-bit AND	and	AND Rd, Rs, \$Rt	\$Rd = \$Rs & \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0100
	andi	ANDI Rt, Rs, imm	\$Rt = \$Rs & imm	001100	5 bits	5 bits	No	16 bits
Bit-by-bit OR	or	OR Rd, Rs, Rt	\$Rd = \$Rs \$Rt	000000	5 bits	5 bits	5 bits	000 0010 0101
	ori	ORI Rt, Rs, imm	\$Rt = \$Rs imm	001101	5 bits	5 bits	No	16 bits
Bit-by-bit NOT	nor	NOR Rd,Rs,Rt	Rd=--(Rt Rt)	000000	5 bits	5 bits	5 bits	000 0010 0111

MIPS	Giải thích		
ori \$t2,\$t1,15			
andi \$t2,\$t1,5			
opcode	Rs (\$t1)	Rt (\$t2)	16 bits (15)
001101	0.1001	0.1010	0000.0000.0000.1111

5. CÁC PHÉP TÍNH LOGIC

Dễ thì sẽ cho dạng gọi mở như sau:

MIPS	opcode	Bit cuối (16bits)
sll	000000	h.000000
srl	000000	h.000010
and	000000	000 0010 0100
andi	001100	16 bits
or	000000	000 0010 0101
ori	001101	16 bits
nor	000000	000 0010 0111

MIPS	Giải thích
srl \$t2,\$t1,5	

srl \$t2,\$t1,5				
opcode	Rs (\$t0)	Rt (\$t1)	Rd (\$t2)	(11 bits) = 5 + ...
000000	0.1000	0.1001	0.1010	00101.000010

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Trong ngôn ngữ lập trình, đưa ra quyết định thường được biểu diễn bằng cách sử dụng câu lệnh "if", đôi khi kết hợp với câu lệnh "go to". ➔ MIPS???

Tiếng anh	opcode	Lệnh ví dụ	Mã giả
branch on equal	000100	beq \$s1, \$s2, 25	if (\$s1 == \$s2) goto PC + 4 + 100
branch on not equal	000101	bne \$s1, \$s2, 25	if (\$s1 != \$s2) goto PC + 4 + 100
set on less than	000000	slt \$s1, \$s2, \$s3	if (\$s2 < \$s3) \$s1 = 1; else \$s1 = 0
set on less than unsigned	000000	sltu \$s1, \$s2, \$s3	if (\$s2 < \$s3) \$s1 = 1; else \$s1 = 0
set on less than immediate unsigned	001011	sltiu \$s1, \$s2, 20	if (\$s2 < 20) \$s1 = 1; else \$s1 = 0
branch on less than	001010	slt \$s1, \$s2, 20	if (\$s2 < 20) \$s1 = 1; else \$s1 = 0
branch greater than	Student	bit	Student
branch less than or equal	Student	bgt	Student
branch greater than or equal	Student	bge	Student

op	Rs	Rt	Constant or address
6 bits	5 bits	5 bits	16 bits

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Ví dụ

MIPS	Giải thích	
bne \$a3,\$a4, Else	# go to Else if i != j	if (\$a3!=\$a4) nhảy đến Else
Something 1	# something 1	if (\$a3==\$a4) thực hiện lệnh ở đây
Something 2	# something 2	
Else: sub \$a0, \$a1, \$a2	# f = g - h (skipped if i = j)	

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Goto in MIPS???

Tiếng anh	Tiếng việt	Lệnh ví dụ	opcode
jum		j label	000010
jump register		jr \$ra	000000
jump and link		jal label	000011

Lệnh	op	address
j label	6 bits	26 bits
jr \$ra	6 bits	5 bits (ra register) 32-11=21 bits
jal label	6 bits	26 bits

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Bài tập: biểu diễn if(a>=b) và if(a<b)

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Biểu diễn if(a>b) và if(a<b)

MIPS	Giải thích
slt \$t0,\$s0,\$s1	# \$t0 = 1 if \$s0 < \$s1 # \$t0 = 0 if \$s0 > \$s1
beq \$t0,\$zero,skip	Beq== \$t0==0 # if \$s0 >= \$s1, goto skip
<stuff>	# do if \$s0 < \$s1
slt \$t0,\$s0,\$s1	# \$t0 = 1 if \$s0 < \$s1
bne \$t0,\$zero,skip	# if \$s0 < \$s1, goto skip
<stuff>	# do if \$s0 >= \$s1

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Biểu diễn if-then-else

if (i == j) f = g + h; else f = g - h;

Biết f , g , h , i và j là các biến. Nếu năm biến f đến j tương ứng với 5 thanh ghi \$s0 đến \$s4, mã MIPS cho câu lệnh *if* này là gì?

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Biểu diễn if-then-else

if (i == j) f = g + h; else f = g - h;

Biết f , g , h , i và j là các biến. Nếu năm biến f đến j tương ứng với 5 thanh ghi \$s0 đến \$s4, mã MIPS cho câu lệnh *if* này là gì?

MIPS	Giải thích
bne \$s3,\$s4,Else	# go to Else if i != j
add \$s0,\$s1,\$s2	# f = g + h (skipped if i != j)
j exit	# go to Exit
Else: sub \$s0,\$s1,\$s2	# f = g - h (skipped if i = j)
exit:	# if \$s0 < \$s1, goto skip

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Biểu diễn while

```
while (save[i] == k)
    i += 1;
```

Giả định rằng i và k tương ứng với thanh ghi \$s3 và \$s5; và địa chỉ nền/cơ sở của mảng **save** lưu trong \$s6. Mã assembly MIPS tương ứng với đoạn mã C trên là gì?

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Biểu diễn while

```
while (save[i] == k)
    i += 1;
```

Giả định rằng i và k tương ứng với thanh ghi \$s3 và \$s5; và địa chỉ nền/cơ sở của mảng **save** lưu trong \$s6. Mã assembly MIPS tương ứng với đoạn mã C trên là gì?

MIPS	Giải thích
Loop: sll \$t1,\$s3,2	# Temp reg \$t1 = 4 * i
add \$t1,\$t1,\$s6	# \$t1 = address of save[i]
lw \$t0,0(\$t1)	# Temp reg \$t0 = save[i]
bne \$t0,\$s5, Exit	# go to Exit if save[i] != k
addi \$s3,\$s3,1	# i = i + 1
j Loop	# go to Loop
Exit:	

6. CÁC LỆNH ĐIỀU KIỆN VÀ NHẢY

❖ Ví dụ: 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? Chuyển sang mã lệnh

```
A[3] = h + A[9];
if(A[3]<8)
    A[3]=A[3]+25;
else
    while(A[3]<100)
        A[3]=A[2]+2;
```
