

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH


Tuần 2:

Instruction Set, Basic Instructions, Directives

Home Assignment 1:

- Em đã đọc nghiên ngẫm :V

Tập thanh ghi của MIPS		
Tên thanh ghi	Số hiệu thanh ghi	Công dụng
\$zero	0	the constant value 0, chứa hằng số = 0
\$at	1	assembler temporary, giá trị tạm thời cho hợp ngữ
\$v0-\$v1	2-3	procedure return values, các giá trị trả về của thủ tục
\$a0-\$a3	4-7	procedure arguments, các tham số vào của thủ tục
\$t0-\$t7	8-15	temporaries, chứa các giá trị tạm thời
\$s0-\$s7	16-23	saved variables, lưu các biến
\$t8-\$t9	24-25	more temporarie, chứa các giá trị tạm thời
\$k0-\$k1	26-27	OS temporaries, các giá trị tạm thời của OS
\$gp	28	global pointer, con trỏ toàn cục
\$sp	29	stack pointer, con trỏ ngăn xếp
\$fp	30	frame pointer, con trỏ khung
\$ra	31	procedure return address, địa chỉ trở về của thủ tục

 NKK-CA2021.1.0 IT3283-Kiến trúc máy tính 26

- Thanh ghi đặc biệt HI và LO, chúng được sử dụng để kiểm soát kết quả của một lệnh thực hiện nhân hoặc chia số nguyên
- PC là thanh ghi chương trình là một thanh ghi được khởi tạo bởi hệ điều hành đến địa chỉ của lệnh đầu tiên của chương trình trong bộ nhớ chú ý rằng địa chỉ trong con trỏ chương trình được gửi đến đầu vào địa chỉ của bộ nhớ thông qua một bus. Sau khi một lệnh được lấy từ bộ nhớ và được nạp vào thanh ghi lệnh IR con trỏ PC sẽ được tăng để CPA có được địa chỉ cho lệnh tuần tự tiếp theo cho quá trình lấy lệnh

- có 3 loại khuôn dạng của 3 lệnh R,I,J

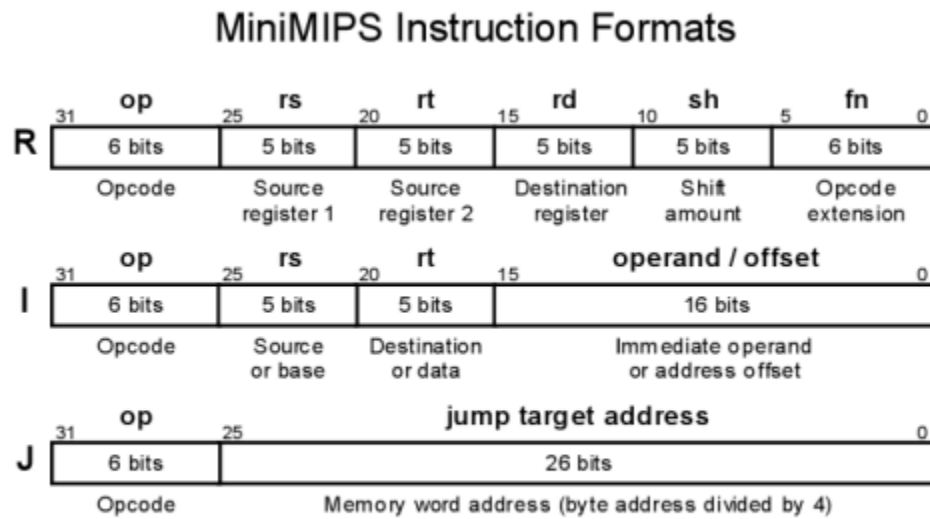


Figure 5.4 MiniMIPS instructions come in only three formats: register (R), immediate (I), and jump (J).

-
-
-

Assignment 1: Lệnh gán số 16-bit

Màn hình code

```

Edit  Execute
Lenhganso16bit.asm
1  #laboratory Exercise 2, Assignment 1
2  .text
3      addi    $s0, $zero, 0x3007    # $s0 = 0 + 0x3007 = 0x3007    I-type
4      add     $s0, $zero, $0        # $s0 = 0 + 0 = 0            R-type
5
6

```

Kết quả chạy

The screenshot shows the MARS MIPS simulator interface. The main window displays assembly code in the 'Text Segment' and its corresponding values in the 'Data Segment'. The 'Registers' window on the right shows the state of the MIPS registers.

Text Segment:

Bkpt	Address	Code	Basic	
	0x00400000	0x20103007	addi \$16,\$0,12295	3: addi \$s0, \$zero, 0x3007
	0x00400004	0x00008020	add \$16,\$0,\$0	4: add \$s0, \$zero, \$0

Data Segment:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0	0	0	0
0x10010020	0	0	0	0
0x10010040	0	0	0	0
0x10010060	0	0	0	0
0x10010080	0	0	0	0
0x100100a0	0	0	0	0
0x100100c0	0	0	0	0
0x100100e0	0	0	0	0
0x10010100	0	0	0	0
0x10010120	0	0	0	0

Registers:

Na...	Nu...	Value
\$...	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	26846...
\$sp	29	21474...
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

Mars Messages:

```

Assemble: assembling C:\Users\Hoang Minh Ngọc\OneDrive - Hanoi University o
Assemble: operation completed successfully.
  
```

Sự thay đổi của \$s0 và \$pc

Ban đầu \$s0 bằng 0, sau lệnh chạy đầu tiên giá trị (Value) = 12295 là câu lệnh

addi \$s0, \$zero, 0x3007 kiểu I câu lệnh cộng dồn $S0 = 0 + 0x3007 = 0x3007$ là mã hexa của 12295 (thập phân)

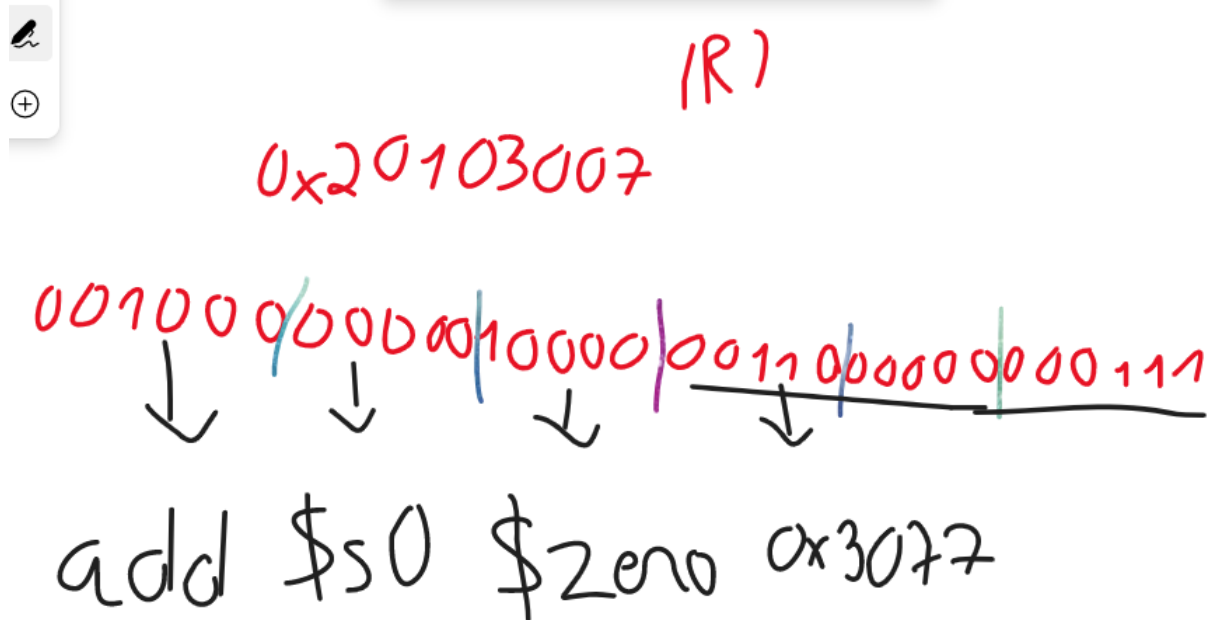
chương trình chạy lệnh số 2 gán add \$s0, \$zero, \$0 kiểu R lệnh gán $S0 = 0 + 0 = 0$ nên giá trị(value) = 0;

sau mỗi 1 lệnh thì thanh PC tăng 4

giá trị khi thay đổi PC = 4194304 -> 4194308 -> 4194312

so sánh mã máy của các lệnh trên khuôn dạng dữ liệu

các dữ liệu đã cho ở bảng trên ta có thể làm phép tính



như vậy mã máy và lệnh trên khuôn hoàn toàn khớp, chứng tỏ tập lệnh đã đúng qui định

sau khi chúng ta chạy lệnh `addi $s0, $zero, 0x2110003d`

cộng dồn `$s0 = 0 + 0x2110003d` thì bản thân số `2110003d` là số 32 bit nên máy sẽ thực hiện cộng từng lệnh tách số `2110003d` ra làm 2

bước 1: thực hiện cộng `$1 = $1 + 0x00002110` lúc này `$1` có giá trị bằng 0)

bước 2: khi đó `$1` có giá trị bằng `0x21100000` ta thực hiện cộng `$1` với `0x0000003d` ta được giá trị là `0x2110003d`

bước 3: cộng kết quả giữa `$0 + $1` lưu vào giá trị ô `$16`

bước 4: thực hiện gán `$16 = 0 + 0 = 0`

kết thúc chương trình vậy bước 1,2,3 sẽ giải quyết câu tính toán khi thay thế lệnh trên

Assignment 2: Lệnh gán số 32-bit

Code

```
Lenhganso16bit.asm  lenhganso32bit.asm
1  #Laboratory Exercise 2, Assignment 2
2  .text
3  lui $s0,0x2110 #put upper half of pattern in $s0
4  ori $s0,0x003d #put lower half of pattern in $s0
```

Kết quả

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x3c102110	lui \$16,0x00002110	3: lui \$s0,0x2110 #put upper ha
<input type="checkbox"/>	0x00400004	0x3610003d	ori \$16,\$16,0x00000...4: ori \$s0,0x003d #put lower ha	

Đầu tiên sẽ cộng dồn $s0 = s0 + 0x2110 = 0x21100000$

Sau đó ta cộng tiếp $s0 = s0 + 0x003d = 0x2110003d$

Sự thay đổi trên thanh ghi \$s0 và \$pc

S0 thay đổi như sự giải thích ở trên, giá trị thanh pc thay đổi theo từng lệnh, cộng 4 giá trị sau mỗi bước

Assignment 3: lệnh gán (giả lệnh)

Code

```

Lenhganso16bit.asm  lenhganso32bit.asm  lenhgan.asm
1  #Laboratory Exercise 2, Assignment 3
2  .text
3  li $s0,0x2110003d #pseudo instruction=2 basic instructions
4  li $s1,0x2 #but if the immediate value is small, one ins
5  |

```

Chạy chương trình

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x3c012110	lui \$1,0x00002110	3: li \$s0,0x2110003d #pseudo ir
<input type="checkbox"/>	0x00400004	0x3430003d	ori \$16,\$1,0x0000003d	
<input type="checkbox"/>	0x00400008	0x24110002	addiu \$17,\$0,0x0000...	4: li \$s1,0x2 #but if the immed

Thực hiện lệnh gán $\$1 = \$1 + 0x00002110$

Thực hiện lệnh gán giá trị $\$s0 = \$s0 + 0x0000003d$

Thực hiện lệnh gán giá trị $\$s1 = \$s1 + 0x2$

Và ta có kết quả sau

\$s0	16	0x2110003d
\$s1	17	0x00000002

Assignment 4: tính giá trị của biểu thức $2x + y = ?$

Code

```

#Laboratory Exercise 2, Assignment 4
.text
# Assign X, Y
addi $t1, $zero, 5 # X = $t1 = ?
addi $t2, $zero, -1 # Y = $t2 = ?
# Expression Z = 2X + Y
add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y

```

Run

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x20090005	addi \$9,\$0,0x00000005	4: addi \$t1, \$zero, 5 # X = \$t1
<input type="checkbox"/>	0x00400004	0x200affff	addi \$10,\$0,0xfffff...	5: addi \$t2, \$zero, -1 # Y = \$t
<input type="checkbox"/>	0x00400008	0x01298020	add \$16,\$9,\$9	7: add \$s0, \$t1, \$t1 # \$s0 = \$
<input type="checkbox"/>	0x0040000c	0x020a8020	add \$16,\$16,\$10	8: add \$s0, \$s0, \$t2 # \$s0 = \$

ở cửa sổ chạy có 4 bước

đầu tiên ta thực hiện gán giá trị $X = 5$ tại thanh ghi $\$t1 = 0x00000005$

tiếp đến ta thực hiện gán $Y = -1$ tại thanh ghi $\$t2 = 0xffffffff$

tại thanh ghi $\$16$ ta thực hiện lệnh cộng $\$9 + \9 tức là $\$t1 + \$t1 = 0x0000000a$

tiếp đến ta cộng dồn $\$16 = \$16 + \$10 = 0x00000009$

vậy khi chuyển từ hexa sang hệ thập phân ta được $= 9$

khi tự tính toán ta nhận được giá trị $2X+Y = 2*5-1 = 9$

phép tính hoàn toàn chính xác

Assignment 5: Phép nhân

Code

```

Leningans0100it.asm                                Leningans0320i
#Laboratory Exercise 2, Assignment 5
.text
# Assign X, Y
addi $t1, $zero, 4 # X = $t1 = ?
addi $t2, $zero, 5 # Y = $t2 = ?
# Expression Z = 3*XY
mul $s0, $t1, $t2 # HI-LO = $t1 * $t2 = X * Y ; $s0 = LO
mul $s0, $s0, 3 # $s0 = $s0 * 3 = 3 * X * Y
# Z' = Z
mflo $s1

```

Run

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x20090004	addi \$9,\$0,0x00000004	4: addi \$t1, \$zero, 4 # X = \$t1
<input type="checkbox"/>	0x00400004	0x200a0005	addi \$10,\$0,0x00000005	5: addi \$t2, \$zero, 5 # Y = \$t2
<input type="checkbox"/>	0x00400008	0x712a8002	mul \$16,\$9,\$10	7: mul \$s0, \$t1, \$t2 # HI-LO = \$s0
<input type="checkbox"/>	0x0040000c	0x20010003	addi \$1,\$0,0x00000003	8: mul \$s0, \$s0, 3 # \$s0 = \$s0 * 3
<input type="checkbox"/>	0x00400010	0x72018002	mul \$16,\$16,\$1	
<input type="checkbox"/>	0x00400014	0x00008812	mflo \$17	10: mflo \$s1

Ta nhận thấy có 6 bước cần thực hiện

Đầu tiên ta thực hiện lệnh cộng $\$9 = 0 + 0x00000004$

Bước tiếp theo ta thực hiện lệnh cộng $\$10 = 0 + 0x00000005$

Bước số 3 lấy phép nhân để nhận kết quả vào thanh ghi lo đồng thời \$16 nhận giá trị bằng $0x00000014$

Do là số có giá trị trong khoảng 0-31 bit nên sẽ được ghi vào thanh ghi lo

Bước số 4 ta cộng $\$1 = \$0 + 0x00000003$

Bước số 5 ta thực hiện phép nhân $\$16 = \$16 * \$1 = 0x0000003c$

Do đã thực hiện 2 phép nhân nên kết quả của phép nhân đầu tiên đang ở lo sẽ được chuyển lên pc và lo nhận giá trị của phép nhân $0x0000003c$

Assignment 6: tạo biến và truy cập biến

Code

```
#Laboratory Exercise 2, Assignment 5
.data # DECLARE VARIABLES
X : .word 5 # Variable X, word type, init value = 5
Y : .word -1 # Variable Y, word type, init value = -1
Z : .word # Variable Z, word type, no init value

.text # DECLARE INSTRUCTIONS
# Load X, Y to registers
la $t8, X # Get the address of X in Data Segment
la $t9, Y # Get the address of Y in Data Segment
lw $t1, 0($t8) # $t1 = X
lw $t2, 0($t9) # $t2 = Y

# Calculate the expression Z = 2X + Y with registers only
add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
# Store result from register to variable Z
la $t7, Z # Get the address of Z in Data Segment
sw $s0, 0($t7) # Z = $s0 = 2X + Y
```


Run

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00000100	8: lui \$t8, X # Get the address of X in I
<input type="checkbox"/>	0x00400004	0x34380000	ori \$24,\$1,0x00000000	Basic assembler instruction
<input type="checkbox"/>	0x00400008	0x3c011001	lui \$1,0x00001001	9: la \$t9, Y # Get the address of Y in I
<input type="checkbox"/>	0x0040000c	0x34390004	ori \$25,\$1,0x00000004	
<input type="checkbox"/>	0x00400010	0x8f090000	lw \$9,0x00000000(\$24)	10: lw \$t1, 0(\$t8) # \$t1 = X
<input type="checkbox"/>	0x00400014	0x8f2a0000	lw \$10,0x00000000(\$...	11: lw \$t2, 0(\$t9) # \$t2 = Y
<input type="checkbox"/>	0x00400018	0x01298020	add \$16,\$9,\$9	13: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 =
<input type="checkbox"/>	0x0040001c	0x020a8020	add \$16,\$16,\$10	14: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 =
<input type="checkbox"/>	0x00400020	0x3c011001	lui \$1,0x00001001	16: la \$t7, Z # Get the address of Z in I
<input type="checkbox"/>	0x00400024	0x342f0008	ori \$15,\$1,0x00000008	
<input type="checkbox"/>	0x00400028	0xadf00000	sw \$16,0x00000000(\$...	17: sw \$s0, 0(\$t7) # Z = \$s0 = 2X + Y

Đầu tiên ta sẽ tạo ra địa chỉ của X tại address 0x10010000 có giá trị bằng 5 (0x00000005) value +4 tức là giá trị của Y và bằng -1 có mã 0xffffffff Z ở vị trí value +8 trên Data Segment

Tiếp đến ta thực hiện lệnh

Trong trường hợp hằng số 32-bit à sử dụng lệnh lui và lệnh ori: lui \$1, 0x00001001 Copy 16 bit cao của hằng số 32-bit vào 16 bit trái của \$1 Xóa 16 bits bên phải của \$1 về 0 ori \$1,\$1,0x00000000 Đưa 16 bit thấp của hằng số 32-bit vào thanh ghi \$1

Tương tự với biến Y ta cũng có như trên

Lệnh lw \$9, 0x00000000 (\$24) ta lấy giá trị của X gán lại vào \$t1

Tương tự như vậy ta gán \$t2 = Y

Ta gán \$s0 bằng \$t1 vì t1 là biến giá trị tạm thời đã được lưu bằng X

Tương tự như Y ta có lời giải thích như ở bài 5

Lệnh lw trong MIPS để load word