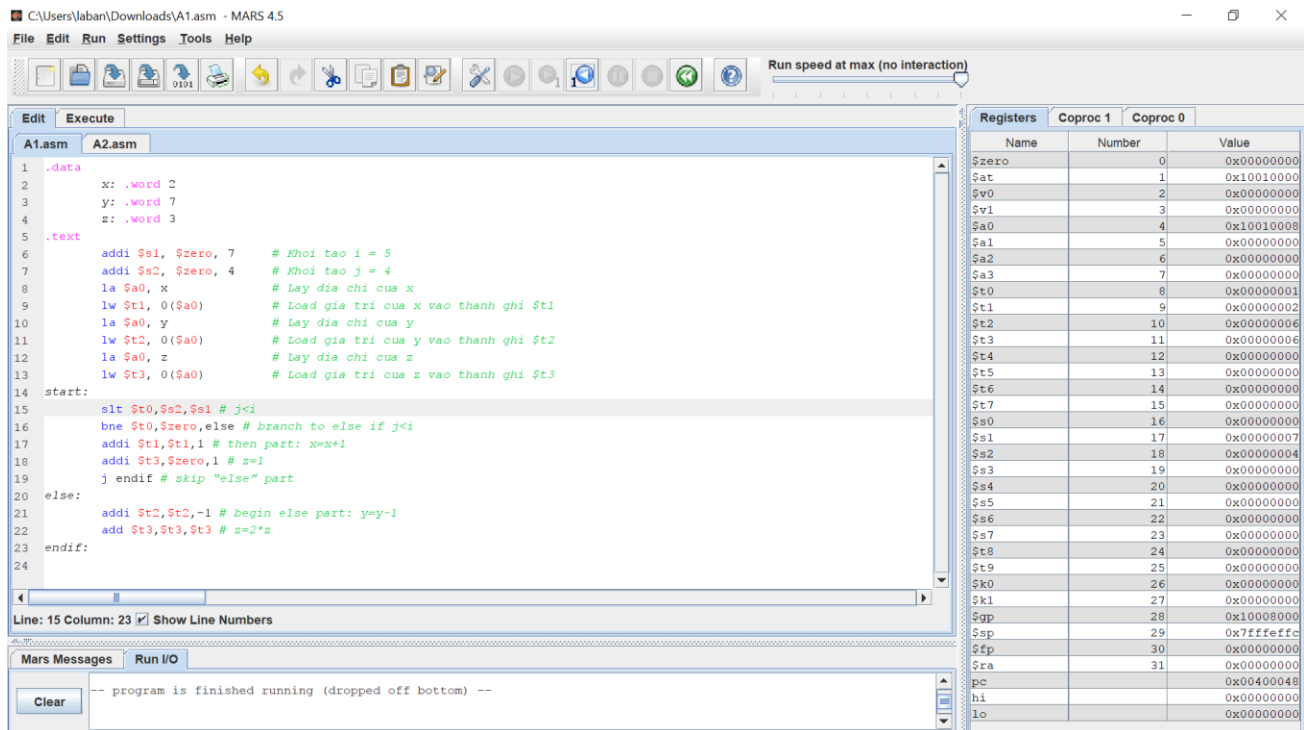


Báo cáo Thực hành KTM T buổi 3

Họ và tên: Nguyễn Đức Phú

MSSV: 20215116

Assignment 1:



- Khởi tạo các giá trị x, y, z lần lượt là 2, 7, 3
- Sau khi chạy các câu lệnh:
`addi $s1, $zero, 7`
`addi $s2, $zero, 4`
⇒ Các thanh ghi `$s1`, `$s2` thay đổi thành các giá trị 7, 4
- Sau khi chạy các câu lệnh:
`la $a0, x`
`lw $t1, 0($a0)`
`la $a0, y`
`lw $t2, 0($a0)`
`la $a0, z`
`lw $t3, 0($a0)`
⇒ Các lệnh `la` đã lấy địa chỉ ô nhớ của biến và lưu vào thanh ghi tạm \$at, lệnh `lw` để ghi giá trị đó vào các thanh ghi `$t1`, `$t2`, `$t3`

- Sau khi chạy câu lệnh:

```
slt $t0, $s2, $s1
```

⇒ Câu lệnh này sẽ thực hiện so sánh $\$s2 < \$s1$ (tức $j < i$), nếu đúng thì thanh ghi $\$t0$ được gán giá trị bằng 1, nếu sai thanh ghi $\$t0$ được gán giá trị bằng 0.

⇒ Sau khi chạy xong câu lệnh ta được kết quả $\$t0 = 1$ do $\$s2 < \$s1$ ($4 < 7$) là đúng

- Sau khi chạy câu lệnh:

```
bne $t0, $zero, else
```

⇒ Câu lệnh so sánh thanh ghi $\$t0$ với thanh ghi $\$zero$, nếu không bằng nhau thì sẽ nhảy đến label else

⇒ Thanh ghi pc lúc này từ giá trị $0x00400030$ nhảy đến else có giá trị $0x00400040$ vì $\$t0 = 0x00000001$ khác $\$zero$

- Sau khi chạy các câu lệnh:

```
addi $t2, $t2, -1
```

```
add $t3, $t3, $t3
```

⇒ Hai câu lệnh trên thực hiện hai phép toán $y = y - 1$ và $z = 2 * z$

⇒ Thanh ghi $\$t2 = 0x00000006$ ($7 - 1 = 6$)

Và $\$t3 = 0x00000006$ ($2 * 3 = 6$)

⇒ Kết quả đúng

- Sau khi chạy câu lệnh:

```
endif
```

⇒ Kết thúc chương trình

• Nếu thay đổi giá trị khởi tạo i, j sao cho $i \leq j$ thì giá trị $\$t0 = 0$

- Tại lệnh: `bne $t0, $zero, else` => thanh ghi pc sẽ không nhảy tới nhãn else mà chỉ tăng thêm 4 và tiếp tục thực hiện các lệnh

```
addi $t1, $t1, 1
```

```
addi $t3, $zero, 1
```

⇒ Thực hiện 2 phép toán $x = x + 1$ và $z = 1$

⇒ Thanh ghi $\$t1 = 0x00000003$ ($2 + 1 = 3$)

Và $\$t3 = 0x00000001$ ($= 1$)

- Tới dòng lệnh `j endif` thì thanh ghi pc nhảy tới nhãn `endif` ($0x00400048$) và kết thúc chương trình

Assignment 2:

- Khởi tạo mảng A gồm 5 phần tử {1,1,-2,-3,5} và $i = 0$, $n = 5$, $step = 1$, $sum = 0$

The screenshot shows the MARS MIPS simulator interface. The main window displays assembly code for a program that initializes an array A with values {1, 1, -2, -3, 5} and sets up variables i, n, step, and sum. The code includes a loop that calculates the sum of the array elements. The right-hand pane shows the register file, with registers \$s1 through \$s5 highlighted in green, indicating they have been initialized to zero. The status bar at the bottom shows 'Line: 23 Column: 1' and 'Show Line Numbers' is checked.

```
1 .data
2     A: .word 1,1,-2,-3,5
3
4 .text
5     addi $s1, $zero, 0    # i = 0
6     addi $s3, $zero, 5    # n = 5
7     addi $s4, $zero, 1    # step = 1
8     addi $s5, $zero, 0    # sum = 0
9     la $s2, A             # load address A[0] to $s2
10
11 loop:
12     slt $t2, $s1, $s3    # $t2 = i < n ? 1 : 0
13     beq $t2, $zero, endloop
14     add $t1, $s1, $s1    # $t1 = 2 * $s1
15     add $t1, $t1, $s2    # $t1 = 4 * $s1
16     add $t1, $t1, $s2    # $t1 store the address of A[i]
17     lw $t0, 0($t1)      # load value of A[i] in $t0
18     add $s5, $s5, $t0    # sum = sum + A[i]
19     add $s1, $s1, $s4    # i = i + step
20     j loop # goto loop
21 endloop:
22
23
24
25
26
27
28
29
```

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000005
\$t1	9	0x10010010
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$s1	17	0x00000005
\$s2	18	0x10010000
\$s3	19	0x00000005
\$s4	20	0x00000001
\$s5	21	0x00000002
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffc00
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0040003e
hi		0x00000000
lo		0x00000000

- Sau khi chạy các câu lệnh:

```
addi $s1, $zero, 0
addi $s3, $zero, 5
addi $s4, $zero, 1
addi $s5, $zero, 0
```

⇒ Các thanh ghi $\$s1$, $\$s3$, $\$s4$, $\$s5$ được gán giá trị lần lượt tương đương với $i=0$, $n=5$, $step=1$, $sum=0$

- Sau khi chạy câu lệnh:

```
la $s2, A
```

⇒ Địa chỉ của mảng A được nạp vào thanh ghi $\$s2$, đây cũng chính là địa chỉ cơ sở A[0]

- Sau khi chạy câu lệnh:

```
slt $t2, $s1, $s3
```

⇒ Câu lệnh kiểm tra $i < n$ không. Vì $\$s1 < \$s3$ nên $\$t2 = 1$

- Sau khi chạy câu lệnh:

```
beq $t2, $zero, endloop
```

⇒ Vì `$t2` khác `$zero` nên thanh ghi `pc` trở đến địa chỉ của câu lệnh tiếp theo mà không nhảy đến nhãn `endloop`

- Sau khi chạy các câu lệnh:

```
add $t1, $s1, $s1
add $t1, $t1, $t1
add $t1, $t1, $s2
```

⇒ Thực hiện các phép toán khiến `$t1 = 4*i` và cộng `$t1` với `$s2` (địa chỉ cơ sở A[0])

⇒ Có được địa chỉ A[i]

- Sau khi chạy lệnh:

```
lw $t0, 0($t1)
```

⇒ Câu lệnh lấy dữ liệu trong ô nhớ có địa chỉ `$t1` và nạp vào `$t0`

- Sau khi chạy các câu lệnh:

```
add $s5, $s5, $t0
add $s1, $s1, $s4
```

⇒ Hai câu lệnh thực hiện tính `sum = sum + A[i]` và `i = i + step`

- Sau khi chạy lệnh:

```
j loop
```

⇒ Câu lệnh nhảy đến nhãn `loop`

- Khi `i = 5`, và câu lệnh `beq` có `$t2 = $zero`, câu lệnh tiếp theo nhảy đến nhãn `endloop` và kết thúc chương trình

- Kết quả `sum = $s5 = 0x00000002` là kết quả đúng của `1+1+(-2)+(-3)+5`

Assignment 3:

The screenshot shows the Mars MIPS simulator interface. The main window displays assembly code for a program that calculates a sum using a loop and a switch statement. The code is as follows:

```
1 .data
2     test: .word 1
3 .text
4     la $s0, test           # load the address of test variable
5     lw $s1, 0($s0)         # load the value of test to register $t1
6     li $t0, 0              # load value for test case
7     li $t1, 1
8     li $t2, 2
9     li $s2, 2
10    li $s3, 1
11    beq $s1, $t0, case_0
12    beq $s1, $t1, case_1
13    beq $s1, $t2, case_2
14    j default
15 case_0:
16     addi $s2, $s2, 1        # a=a+1
17     j continue
18 case_1:
19     sub $s2, $s2, $t1       # a=a-1
20     j continue
21 case_2:
22     add $s3, $s3, $s3       # b=2*b
23     j continue
24 default:
25     continue:
```

The right panel shows the register file with the following values:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000001
\$t2	10	0x00000002
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x10010000
\$s1	17	0x00000001
\$s2	18	0x00000001
\$s3	19	0x00000001
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffc00
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400048
hi		0x00000000
lo		0x00000000

The bottom panel shows the message window with the text: "-- program is finished running (dropped off bottom) --".

Dữ liệu của *test* được ghi vào *\$s1*, so sánh *\$s1* với *\$t0*, *\$t1*, *\$t2*

\$s1 = *\$t0* => nhảy tới *case_0*

\$s1 = *\$t1* => nhảy tới *case_1*

\$s1 = *\$t2* => nhảy tới *case_2*

Khi cả ba đều không bằng *\$s1* khi chạy đến lệnh *j default* thì lệnh sẽ nhảy đến nhãn *default*

Trường hợp này *\$s1* = *\$t1* = 1 nên sẽ thực hiện lệnh trong *case_1*

⇒ Kết quả thu được: *\$s2* = *\$s2* - 1 = 2 - 1 = 1

Với trường hợp *\$s1* = *\$t0* = 0 => Thực hiện *case_0*

⇒ Kết quả: *\$s2* = *\$s2* + 1 = 2 + 1 = 3

Với trường hợp *\$s1* = *\$t2* = 2 => Thực hiện *case_2*

⇒ Kết quả: *\$s3* = *\$s3* + *\$s3* = 1 + 1 = 2

- Sau khi thực hiện các case thì tới lệnh *j continue*, nhảy tới nhãn *continue* và kết thúc chương trình

Với trường hợp *\$s1* không bằng giá trị nào trong 3 giá trị *\$t0*, *\$t1*, *\$t2*

⇒ Không nhảy tới case nào, pc tiếp tục tăng 4 tới *j default* nhảy tới nhãn *default*, tiếp tục tăng 4 tới *continue* và kết thúc chương trình

Assignment 4:

a. $i < j$

.data

```
x: .word 2
y: .word 7
z: .word 3
```

.text

```
addi $s1, $zero, 7
addi $s2, $zero, 4
la $a0, x
lw $t1, 0($a0)
la $a0, y
lw $t2, 0($a0)
la $a0, z
lw $t3, 0($a0)
```

start:

```

    slt $t0,$s1,$s2 # i<j
    beq $t0,$zero,else # branch to else if i>=j
    addi $t1,$t1,1 # then part: x=x+1
    addi $t3,$zero,1 # z=1
    j endif # skip "else" part
else:
    addi $t2,$t2,-1 # begin else part: y=y-1
    add $t3,$t3,$t3 # z=2*z
endif:

```

b. $i \geq j$

```

.data
x: .word 2
y: .word 7
z: .word 3

.text
addi $s1, $zero, 7
addi $s2, $zero, 4
la $a0, x
lw $t1, 0($a0)
la $a0, y
lw $t2, 0($a0)
la $a0, z
lw $t3, 0($a0)

start:
    slt $t0,$s1,$s2 # i<j
    bne $t0,$zero,else # branch to else if i<j
    addi $t1,$t1,1 # then part: x=x+1
    addi $t3,$zero,1 # z=1
    j endif # skip "else" part
else:
    addi $t2,$t2,-1 # begin else part: y=y-1
    add $t3,$t3,$t3 # z=2*z
endif:

```

c. $i + j \leq 0$

```
.data
    x: .word 2
    y: .word 7
    z: .word 3

.text
    addi $s1, $zero, 7
    addi $s2, $zero, 4
    la $a0, x
    lw $t1, 0($a0)
    la $a0, y
    lw $t2, 0($a0)
    la $a0, z
    lw $t3, 0($a0)

start:
    add $t4, $s1,$s2 # $t4=i+j
    sgt $t0,$t4,0      # $t4 > 0
    bne $t0,$zero,else # branch to else if i+j >0
    addi $t1,$t1,1 # then part: x=x+1
    addi $t3,$zero,1 # z=1
    j endif # skip "else" part

else:
    addi $t2,$t2,-1 # begin else part: y=y-1
    add $t3,$t3,$t3 # z=2*z

endif:
```

d. $i + j = m + n$

```
.data
    x: .word 2
    y: .word 7
    z: .word 3

.text
    addi $s1, $zero, 7
    addi $s2, $zero, 4
```

```

    la $a0, x
    lw $t1, 0($a0)
    la $a0, y
    lw $t2, 0($a0)
    la $a0, z
    lw $t3, 0($a0)
    addi $s5,$zero,1 #m=1
    addi $s6,$zero,2 #n=2
start:
    add $t4, $s1,$s2 # $t4=i+j
    add $t5, $s5,$s6 # $t5=m+n
    slt $t0,$t5,$t4 # $t5 < $t4
    beq $t0,$zero,else # go to else if m+n >= i+j
    addi $t1,$t1,1 # then part: x=x+1
    addi $t3,$zero,1 # z=1
    j endif # skip "else" part
else:
    addi $t2,$t2,-1 # begin else part: y=y-1
    add $t3,$t3,$t3 # z=2*z
endif:

```

Assignment 5:

- $i < n$ (đã làm)
- $i \leq n$

.data

```
A: .word 1,1,-2,-3,5
```

.text

```

addi $s1, $zero, 0    # i = 0
addi $s3, $zero, 5    # n = 5
addi $s4, $zero, 1    # step = 1
addi $s5, $zero, 0    # sum = 0
la $s2, A              # load address A[0] to $s2

```

loop:

```

slt $t2, $s3, $s1 # $t2 = n < i ? 1 : 0
bne $t2, $zero, endloop # go to end if i>n
add $t1, $s1, $s1 # $t1 = 2 * $s1

```



```

    add $t1, $t1, $t1 # $t1 = 4 * $s1
    add $t1, $t1, $
    lw $t0, 0($t1)
    add $s5, $s5, $t0 # sum = sum + A[i]
    add $s1, $s1, $s4 # i = i + step
    j loop # goto loop

```

endloop:

c. $\text{sum} \geq 0$

.data

```

A: .word 1,1,-2,-3,5

```

.text

```

    addi $s1, $zero, 0    # i = 0
    addi $s3, $zero, 5    # n = 5
    addi $s4, $zero, 1    # step = 1
    addi $s5, $zero, 0    # sum = 0
    la $s2, A             # load address A[0] to $s2

```

loop:

```

slt $t2, $s5, $zero # $t2 = sum < 0 ? 1 : 0
bne $t2, $zero, endloop # go to end if sum < 0
    add $t1, $s1, $s1 # $t1 = 2 * $s1
    add $t1, $t1, $t1 # $t1 = 4 * $s1
    add $t1, $t1, $s2
    lw $t0, 0($t1)
    add $s5, $s5, $t0 # sum = sum + A[i]
    add $s1, $s1, $s4 # i = i + step
    j loop # goto loop

```

endloop:

d. $A[i] == 0$

.data

```

A: .word 1,1,-2,0,5

```

.text

```

    addi $s1, $zero, 0    # i = 0
    addi $s3, $zero, 5    # n = 5

```

```

addi $s4, $zero, 1    # step = 1
addi $s5, $zero, 0    # sum = 0
la $s2, A              # load address A[0] to $s2

```

loop:

```

slt $t2, $s1, $s3 # $t2 = i < n ? 1 : 0
beq $t2, $zero, endloop
add $t1, $s1, $s1 # $t1 = 2 * $s1
add $t1, $t1, $t1 # $t1 = 4 * $s1
add $t1, $t1, $s2 # $t1 store the address of A[i]
lw $t0, 0($t1) # $t0=A[i]
beq $t0,$zero, endloop #if A[i]=0 => end
add $s5, $s5, $t0 # sum = sum + A[i]
add $s1, $s1, $s4 # i = i + step
j loop # goto loop

```

endloop:

Assignment 6:

`.data`

```
A: .word 1, -3, -10, 6, -29, 3, -39
message: .ascii "Tri tuyet doi lon nhat la: "
```

`.text`

```
addi $s0, $zero, 0    # max = 0
la $a0, A
lw $s1, 0($a0)        # A[0]
addi $s2, $zero, 0    # i = 0
addi $s3, $zero, 7    # n = 7
```

```
loop: slt $t2, $s2, $s3 # i < n
      beq $t2, $zero, endloop # i >= n branch to endloop
      sll $t1, $s2, 2      # t1 = i * 4
      add $a1, $a0, $t1 # a1 = a0 + 4
      lw $s4, 0($a1)      # s4 = A[i]
```

`if_nhohon_0:`

```
bgez $s4, if_lonhon_0 # s4 > 0 branch to if_lonhon
sub $s4, $zero, $s4   # s4 = 0 - s4
j if_lonhon_0
```

`if_lonhon_0:`

```
slt $t4, $s0, $s4 # max < s4
bne $t4, $zero, max # max < s4 branch to max
j reloop          # jump reloop
```

```
max: add $s0, $zero, $s4 # max = 0 + s4
      j reloop          # jump reloop
```

`reloop:`

```
addi $s2, $s2, 1      # i = i + 1
j loop                # jump loop
```

`endloop:`