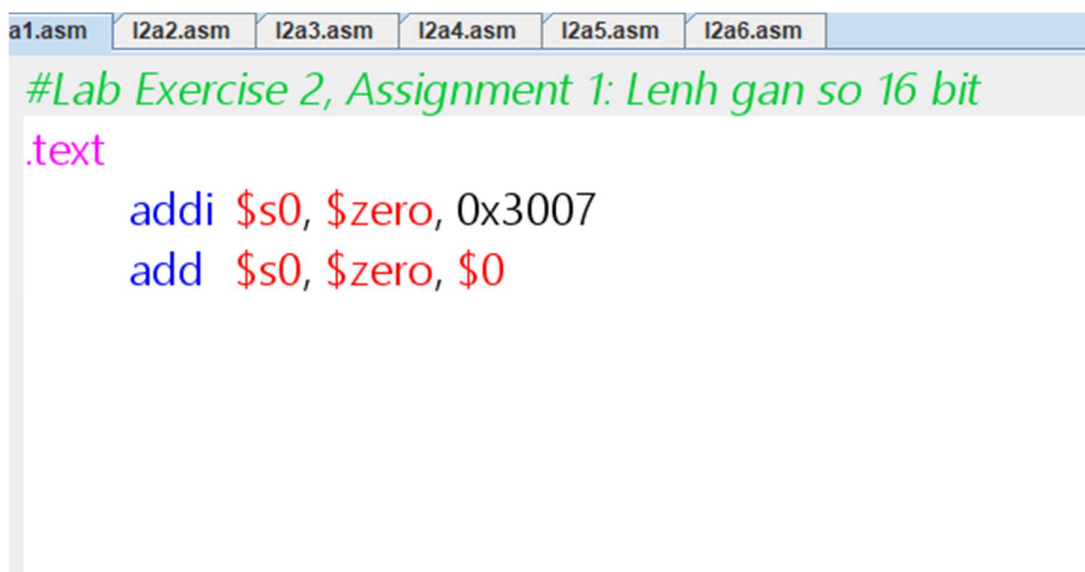


# Báo cáo thực hành KTMT tuần 2

Họ và tên: Đỗ Gia Huy

MSSV: 20215060

## Assignment 1



```
a1.asm l2a2.asm l2a3.asm l2a4.asm l2a5.asm l2a6.asm
#Lab Exercise 2, Assignment 1: Lenh gan so 16 bit
.text
    addi $s0, $zero, 0x3007
    add  $s0, $zero, $0
```

-Sự thay đổi của thanh ghi \$s0: từ giá trị 0x00000000 chuyển thành 0x00003007 sau lệnh thứ nhất, rồi trở lại giá trị ban đầu sau lệnh thứ hai

-Sự thay đổi của thanh ghi pc: tăng thêm một khoảng có giá trị là 0x00000004 sau mỗi giá trị

-So sánh mã máy:

+Lệnh 1:

Là lệnh I, opcode: 8 => 001000, rs: 0 => 00000, rt: 16 => 10000, imm: 0x3007 = 0011 0000 0000 0111

Vậy lệnh máy là: 0010 0000 0001 0000 0011 0000 0000 0111

+Lệnh 2:

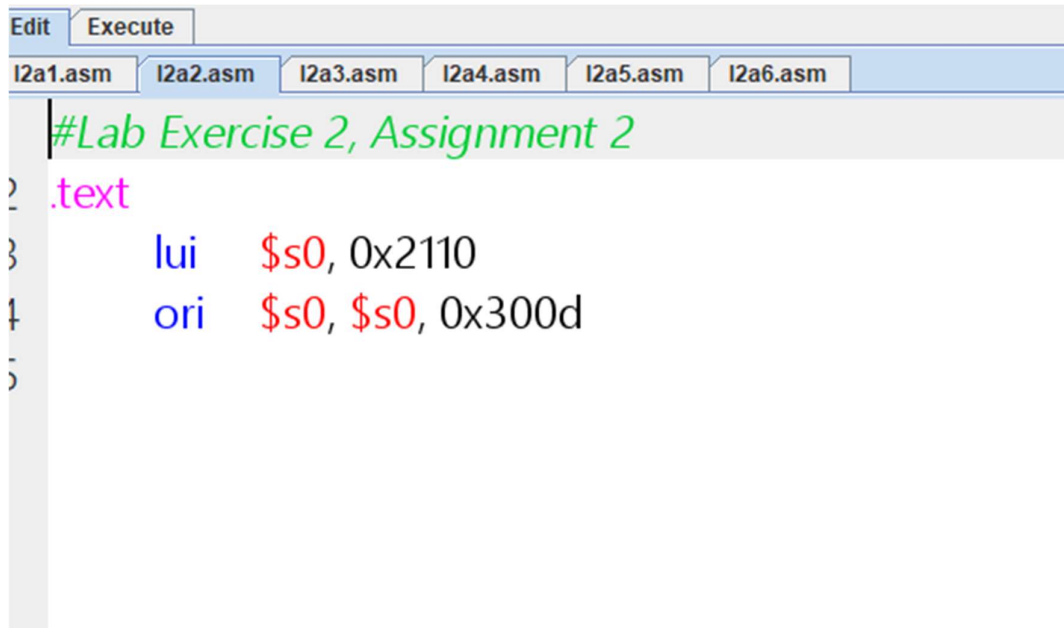
Là lệnh R, opcode: 0 => 000000, rs: 0 => 00000, rt: 0 => 00000, rd: 16 => 10000, sh: 0 => 00000, fn: 32 => 100000

Vậy lệnh máy là: 0000 0000 0000 0000 1000 0000 0010 0000

⇒ Kết quả giống như đang chạy trên ứng dụng

- Nếu sửa lệnh thứ 2 thành `addi $s0, $zero, 0x2110003d` thì nó thành lệnh gán 32 bit, vì số `0x2110003d` cần lưu trữ ở dạng 32 bit

## Assignment 2



```
Edit Execute
l2a1.asm l2a2.asm l2a3.asm l2a4.asm l2a5.asm l2a6.asm
#Lab Exercise 2, Assignment 2
2 .text
3     lui    $s0, 0x2110
4     ori    $s0, $s0, 0x300d
5
```

- Sự thay đổi của thanh ghi `$s0`: từ giá trị `0x00000000` chuyển thành `0x21100000` sau lệnh thứ nhất, rồi biến thành giá trị ban `0x2110300d` sau lệnh thứ hai
- Sự thay đổi của thanh ghi `pc`: tăng thêm một khoảng có giá trị là `0x00000004` sau mỗi giá trị
- Các byte đầu tiên ở vùng lệnh trùng với cột Code (Mã máy theo Hexa) trong cửa sổ Text Segment ở phần thực thi

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c102110	lui \$16, 0x00002110	3: lui \$s0, 0x2110
	0x00400004	0x3610300d	ori \$16, \$16, 0x0000300d	4: ori \$s0, \$s0, 0x300d

Labels

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00400000	0x3c102110	0x3610300d	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x00400000 (.text) Hexadecimal Addresses Hexadecimal Values ASCII

## Assignment 3

Edit Execute

Text editor for composing MIPS programs. 2a4.asm l2a5.asm l2a6.asm

```
#Lab Exercise 3, Assignment 3
2 .text
3     li    $s0, 0x2110003d
4     li    $s1, 0x2
5
```

⇒ Kết quả thực thi các lệnh trên:

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

-Có điều bất thường xảy ra!

-Điều bất thường đó là lệnh li thứ nhất tự tách ra thành 2 lệnh đó là lệnh lui và lệnh ori, còn lệnh li thứ hai bị biến thành lệnh addiu

-Giải thích: Lệnh li gán giá trị số nguyên bất kỳ, trong trường hợp bài Assignment, lệnh li thứ nhất lệnh li thực hiện gán số 0x2110003d là số loại 32 bit nên tự động tách thành 2 lệnh như trên, còn lệnh li thứ hai thực hiện gán số 0x2 là loại số 16 bit không dấu nên chuyển thành lệnh addiu.

## Assignment 4

```

2a1.asm  2a2.asm  2a3.asm  2a4.asm  2a5.asm  2a6.asm
#Lab Exercise 2, Assignment 4
.text
    #Cho gia tri cua 2 thanh ghi $t0, $t1
    addi $t1, $zero, 5          #Bien X o thanh ghi $t1
    addi $t2, $zero, -1         #Bien Y o thanh ghi $t2
    #Tinh bieu thuc Z = 2X+Y, luu vao thanh ghi $s0
    add  $s0, $t1, $t1          #Z=X+X=2X
    add  $s0, $s0, $t2          #Z=Z+Y=2X+Y

```

-Có sự thay đổi của các thanh ghi: \$t1 có giá trị trở thành 0x00000005, \$t2 có giá trị trở thành 0xffffffff, \$s0 thay đổi theo như kết quả Z khi thực hiện phép tính đã được giải thích như trong mã nguồn trên, còn thanh ghi pc cứ mỗi một lệnh tăng thêm 0x00000004.

=> **Kết quả thực thi:**

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

-Từ lệnh addi, điểm tương đồng với hợp ngữ và mã máy là: Sau khi chuyển các yếu tố trong lệnh loại I (addi là loại I) như opcode, rs, rt, imm thành giá trị nhị phân, và ghép lần lượt các giá trị nhị phân lần lượt: opcode-rs-rt-imm thì kết quả ra lệnh mã máy là số nhị phân 32 bit. Khi chuyển lệnh mã máy đó sang Hexa thì kết quả giống hệt như trong cột Code và dòng lệnh tương ứng như trên. Do đó nó là điểm tương đồng.

-Kiểm nghiệm các khuôn mẫu của 4 lệnh như bài 4:

addi \$t1, \$zero, 5

addi \$s9, \$0, 0x5

opcode: 8 => 001000

rs: \$0 => 00000

rt: \$9 => 01001

imm: 0x5 => 0000 0000 0000 0101

0010 0000 0000 1001 0000 0000 0000 0101

0x20090005

addi \$t2, \$zero, -1

addi \$10, \$0, 0xffffffff

opcode: 8 => 001000

rs: \$0 => 00000

rt: \$10 => 01010

imm 0xffffffff => 1111 1111 1111 1111

0010 0000 0000 1010 1111 1111 1111 1111

0x200affff

0x01298020

0000 0001 0010 1001 1000 0000 0010 0000

Opcode: 000000 => 0

rs: 01001 => \$9

rt: 01001 => \$9

rd: 10000 => \$16

sh: 00000

fn: 100000

add \$16, \$9, \$9

0x020a8020

0000 0010 0000 1010 1000 0000 0010 0000

Opcode: 000000 => 0

rs: 10000 => \$16

rt: 01010 => \$10

rd: 10000 => \$16

sh: 00000 fn:

100000 add \$16, \$16, \$10

## Assignment 5

```
l2a1.asm l2a2.asm l2a3.asm l2a4.asm l2a5.asm l2a6.asm
1  #Lab Exercise 2, Assignment 5
2  .text
3      #Gan bien
4      addi $t1, $zero, 4    #Bien X vao thanh ghi $t1
5      addi $t2, $zero, 5    #Bien Y vao thanh ghi $t2
6      #Tinh Z = 3*X*Y
7      mul  $s0, $t1, $t2    #Z=X*Y
8      mul  $s0, $s0, 3      #Z=Z*3 = 3*Z = 3*X*Y
9      #Z' = Z
10     mflo $s1
11
```

=> Kết quả thực thi các lệnh trên:

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

Data Segment				
--------------	--	--	--	--

-Có điều bất thường xảy ra!

-Lệnh mul thứ hai sẽ bị tách thành lệnh addi và lệnh mul. Bởi vì lệnh mul không hỗ trợ việc nhân thanh ghi với một số, nên nó phải tách thành lệnh addi để lưu 1 số vào 1 thanh ghi và dùng lệnh mul sau khi tách để nhân 2 thanh ghi đó với nhau.

-Sự thay đổi các thanh ghi: thanh ghi \$t1, \$t2 dùng để gán giá trị 4 và 5, thanh ghi lo thay đổi thành 0x0000003c, thanh ghi hi không thay đổi, các thanh ghi thay đổi như trên chú thích của mã nguồn trên. Còn thanh ghi at lưu giá trị tạm thời 0x00000003 khi thực hiện phép nhân với số 3.

-Kết quả các thanh ghi sau khi kết thúc chương trình:



\$zero	0	0x00000000
\$at	1	0x00000003
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x0000003c
\$s1	17	0x0000003c
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400018
hi		0x00000000
lo		0x0000003c

⇒ Kết quả đó đúng như lý thuyết.

## Assignment 6



```

1  I2a1.asm  I2a2.asm  I2a3.asm  I2a4.asm  I2a5.asm  I2a6.asm
2  .data      # DECLARE VARIABLES
3  X: .word   5      # Variable X, word type, init value =
4  Y: .word  -1      # Variable Y, word type, init value =
5  Z: .word           # Variable Z, word type, no init value
6  .text      # DECLARE INSTRUCTIONS
7  # Load X, Y to registers
8  la $t8, X      # Get the address of X in Data Segment
9  la $t9, Y      # Get the address of Y in Data Segment
10 lw $t1, 0($t8)  # $t1 = X
11 lw $t2, 0($t9)  # $t2 = Y
12
13 # Calculate the expression Z = 2X + Y with registers only
14 add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
15 add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
16

```

-Lệnh la được biên dịch thành 2 lệnh lui và ori, có tác dụng gán 1 số 32 bit vào thanh ghi

-Khi biên dịch lệnh la thành mã máy, các biến X, Y, Z với hằng số là bằng nhau.

-Khi thực thi chương trình:

The screenshot shows the MARS MIPS simulator interface. The main window displays the assembly code from the file I2a6.asm. The code is as follows:

```

1: lui $t8, 0x10010000 # Get the address of X
2: ori $t8, $t8, 0x00000000 # Get the address of X
3: lui $t9, 0x10010004 # Get the address of Y
4: ori $t9, $t9, 0x00000000 # Get the address of Y
5: lw $t1, 0($t8) # $t1 = X
6: lw $t2, 0($t9) # $t2 = Y
7:
8: add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
9: add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
10:

```

The Labels window shows the following labels and addresses:

Label	Address
X	0x10010000
Y	0x10010004
Z	0x10010008

The Registers window shows the following registers and values:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$a5	21	0x00000000
\$a6	22	0x00000000
\$a7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$t0	26	0x00000000
\$t1	27	0x00000000
\$t2	28	0x00000000
\$t3	29	0x00000000
\$t4	30	0x00000000
\$t5	31	0x00000000
\$t6	32	0x00000000
\$t7	33	0x00000000
\$t8	34	0x00000000
\$t9	35	0x00000000
\$t0	36	0x00000000
\$t1	37	0x00000000
\$t2	38	0x00000000
\$t3	39	0x00000000
\$t4	40	0x00000000
\$t5	41	0x00000000
\$t6	42	0x00000000
\$t7	43	0x00000000
\$t8	44	0x00000000
\$t9	45	0x00000000
\$t0	46	0x00000000
\$t1	47	0x00000000
\$t2	48	0x00000000
\$t3	49	0x00000000
\$t4	50	0x00000000
\$t5	51	0x00000000
\$t6	52	0x00000000
\$t7	53	0x00000000
\$t8	54	0x00000000
\$t9	55	0x00000000
\$t0	56	0x00000000
\$t1	57	0x00000000
\$t2	58	0x00000000
\$t3	59	0x00000000
\$t4	60	0x00000000
\$t5	61	0x00000000
\$t6	62	0x00000000
\$t7	63	0x00000000
\$t8	64	0x00000000
\$t9	65	0x00000000
\$t0	66	0x00000000
\$t1	67	0x00000000
\$t2	68	0x00000000
\$t3	69	0x00000000
\$t4	70	0x00000000
\$t5	71	0x00000000
\$t6	72	0x00000000
\$t7	73	0x00000000
\$t8	74	0x00000000
\$t9	75	0x00000000
\$t0	76	0x00000000
\$t1	77	0x00000000
\$t2	78	0x00000000
\$t3	79	0x00000000
\$t4	80	0x00000000
\$t5	81	0x00000000
\$t6	82	0x00000000
\$t7	83	0x00000000
\$t8	84	0x00000000
\$t9	85	0x00000000
\$t0	86	0x00000000
\$t1	87	0x00000000
\$t2	88	0x00000000
\$t3	89	0x00000000
\$t4	90	0x00000000
\$t5	91	0x00000000
\$t6	92	0x00000000
\$t7	93	0x00000000
\$t8	94	0x00000000
\$t9	95	0x00000000
\$t0	96	0x00000000
\$t1	97	0x00000000
\$t2	98	0x00000000
\$t3	99	0x00000000
\$t4	100	0x00000000
\$t5	101	0x00000000
\$t6	102	0x00000000
\$t7	103	0x00000000
\$t8	104	0x00000000
\$t9	105	0x00000000
\$t0	106	0x00000000
\$t1	107	0x00000000
\$t2	108	0x00000000
\$t3	109	0x00000000
\$t4	110	0x00000000
\$t5	111	0x00000000
\$t6	112	0x00000000
\$t7	113	0x00000000
\$t8	114	0x00000000
\$t9	115	0x00000000
\$t0	116	0x00000000
\$t1	117	0x00000000
\$t2	118	0x00000000
\$t3	119	0x00000000
\$t4	120	0x00000000
\$t5	121	0x00000000
\$t6	122	0x00000000
\$t7	123	0x00000000
\$t8	124	0x00000000
\$t9	125	0x00000000
\$t0	126	0x00000000
\$t1	127	0x00000000
\$t2	128	0x00000000
\$t3	129	0x00000000
\$t4	130	0x00000000
\$t5	131	0x00000000
\$t6	132	0x00000000
\$t7	133	0x00000000
\$t8	134	0x00000000
\$t9	135	0x00000000
\$t0	136	0x00000000
\$t1	137	0x00000000
\$t2	138	0x00000000
\$t3	139	0x00000000
\$t4	140	0x00000000
\$t5	141	0x00000000
\$t6	142	0x00000000
\$t7	143	0x00000000
\$t8	144	0x00000000
\$t9	145	0x00000000
\$t0	146	0x00000000
\$t1	147	0x00000000
\$t2	148	0x00000000
\$t3	149	0x00000000
\$t4	150	0x00000000
\$t5	151	0x00000000
\$t6	152	0x00000000
\$t7	153	0x00000000
\$t8	154	0x00000000
\$t9	155	0x00000000
\$t0	156	0x00000000
\$t1	157	0x00000000
\$t2	158	0x00000000
\$t3	159	0x00000000
\$t4	160	0x00000000
\$t5	161	0x00000000
\$t6	162	0x00000000
\$t7	163	0x00000000
\$t8	164	0x00000000
\$t9	165	0x00000000
\$t0	166	0x00000000
\$t1	167	0x00000000
\$t2	168	0x00000000
\$t3	169	0x00000000
\$t4	170	0x00000000
\$t5	171	0x00000000
\$t6	172	0x00000000
\$t7	173	0x00000000
\$t8	174	0x00000000
\$t9	175	0x00000000
\$t0	176	0x00000000
\$t1	177	0x00000000
\$t2	178	0x00000000
\$t3	179	0x00000000
\$t4	180	0x00000000
\$t5	181	0x00000000
\$t6	182	0x00000000
\$t7	183	0x00000000
\$t8	184	0x00000000
\$t9	185	0x00000000
\$t0	186	0x00000000
\$t1	187	0x00000000
\$t2	188	0x00000000
\$t3	189	0x00000000
\$t4	190	0x00000000
\$t5	191	0x00000000
\$t6	192	0x00000000
\$t7	193	0x00000000
\$t8	194	0x00000000
\$t9	195	0x00000000
\$t0	196	0x00000000
\$t1	197	0x00000000
\$t2	198	0x00000000
\$t3	199	0x00000000
\$t4	200	0x00000000
\$t5	201	0x00000000
\$t6	202	0x00000000
\$t7	203	0x00000000
\$t8	204	0x00000000
\$t9	205	0x00000000
\$t0	206	0x00000000
\$t1	207	0x00000000
\$t2	208	0x00000000
\$t3	209	0x00000000
\$t4	210	0x00000000
\$t5	211	0x00000000
\$t6	212	0x00000000
\$t7	213	0x00000000
\$t8	214	0x00000000
\$t9	215	0x00000000
\$t0	216	0x00000000
\$t1	217	0x00000000
\$t2	218	0x00000000
\$t3	219	0x00000000
\$t4	220	0x00000000
\$t5	221	0x00000000
\$t6	222	0x00000000
\$t7	223	0x00000000
\$t8	224	0x00000000
\$t9	225	0x00000000
\$t0	226	0x00000000
\$t1	227	0x00000000
\$t2	228	0x00000000
\$t3	229	0x00000000
\$t4	230	0x00000000
\$t5	231	0x00000000
\$t6	232	0x00000000
\$t7	233	0x00000000
\$t8	234	0x00000000
\$t9	235	0x00000000
\$t0	236	0x00000000
\$t1	237	0x00000000
\$t2	238	0x00000000
\$t3	239	0x00000000
\$t4	240	0x00000000
\$t5	241	0x00000000
\$t6	242	0x00000000
\$t7	243	0x00000000
\$t8	244	0x00000000
\$t9	245	0x00000000
\$t0	246	0x00000000
\$t1	247	0x00000000
\$t2	248	0x00000000
\$t3	249	0x00000000
\$t4	250	0x00000000
\$t5	251	0x00000000
\$t6	252	0x00000000
\$t7	253	0x00000000
\$t8	254	0x00000000
\$t9	255	0x00000000
\$t0	256	0x00000000
\$t1	257	0x00000000
\$t2	258	0x00000000
\$t3	259	0x00000000
\$t4	260	0x00000000
\$t5	261	0x00000000
\$t6	262	0x00000000
\$t7	263	0x00000000
\$t8	264	0x00000000
\$t9	265	0x00000000
\$t0	266	0x00000000
\$t1	267	0x00000000
\$t2	268	0x00000000
\$t3	269	0x00000000
\$t4	270	0x00000000
\$t5	271	0x00000000
\$t6	272	0x00000000
\$t7	273	0x00000000
\$t8	274	0x00000000
\$t9	275	0x00000000
\$t0	276	0x00000000
\$t1	277	0x00000000
\$t2	278	0x00000000
\$t3	279	0x00000000
\$t4	280	0x00000000
\$t5	281	0x00000000
\$t6	282	0x00000000
\$t7	283	0x00000000
\$t8	284	0x00000000
\$t9	285	0x00000

-Các thanh ghi bị thay đổi và có thay đổi khi chạy chương trình là: \$t1, \$t2, \$t7, \$t8, \$t9, \$s0, at (ghi vào bộ nhớ tạm) và pc (luôn tăng 0x00000004 sau mỗi lệnh)

-Lệnh lw: Lấy địa chỉ của biến kiểu word và lưu vào 1 thanh ghi

-Lệnh sw: Lấy địa chỉ của biến kiểu word lưu vào bộ nhớ