

Assembly Language and Computer Architecture Lab

Week 2

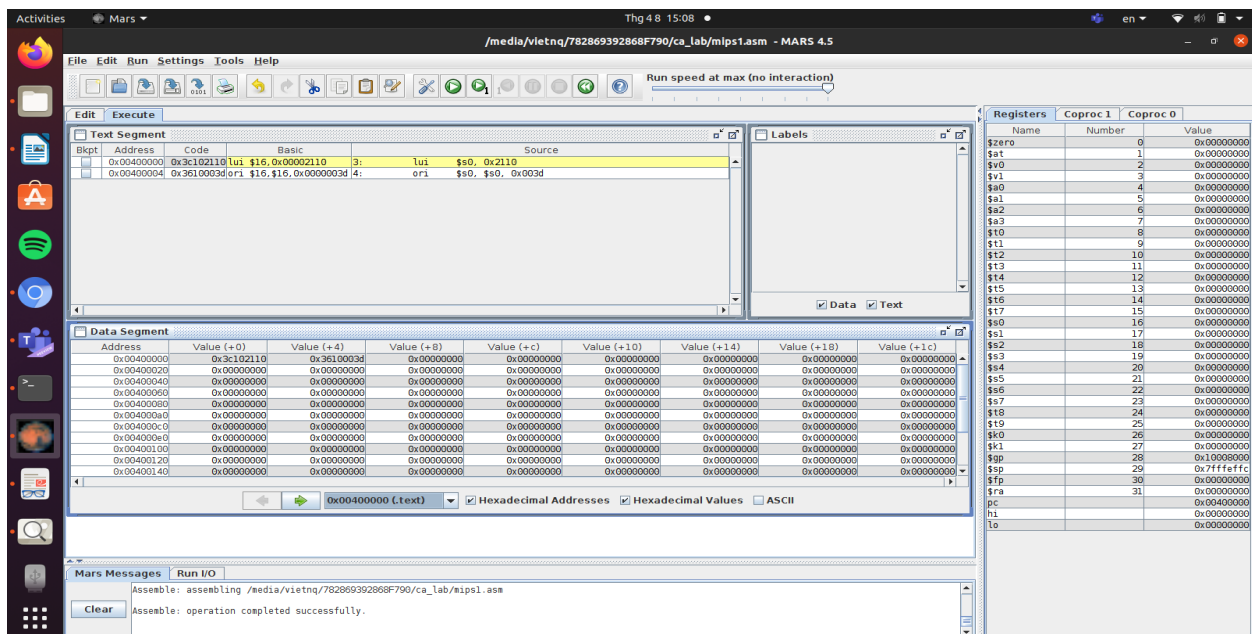
Assignment 1: Assigning a 16-bit number to a register

Source code: **addi \$s0, \$zero, 0x2110003d** is to calculate sum of 2 values **0x2110003d** and **0**, then store the final value to register **\$s0**.

MIPS implemented 3 instructions:

- Set high-order 16 bits of **\$at** to **0x2110** and low-order 16 bits to 0
- Use bitwise OR immediate to set **\$at** to bitwise OR of itself and **0x003d** (value of **0x2110003d** is now stored in **\$at**).
- Add **\$zero** to **\$at**, then store final result to **\$s0**.

Assignment 2: Assigning a 32-bit number to a register



The location (address) of each instruction as we can see in the first row of the Data Segment table, they are at the first 2 cells:

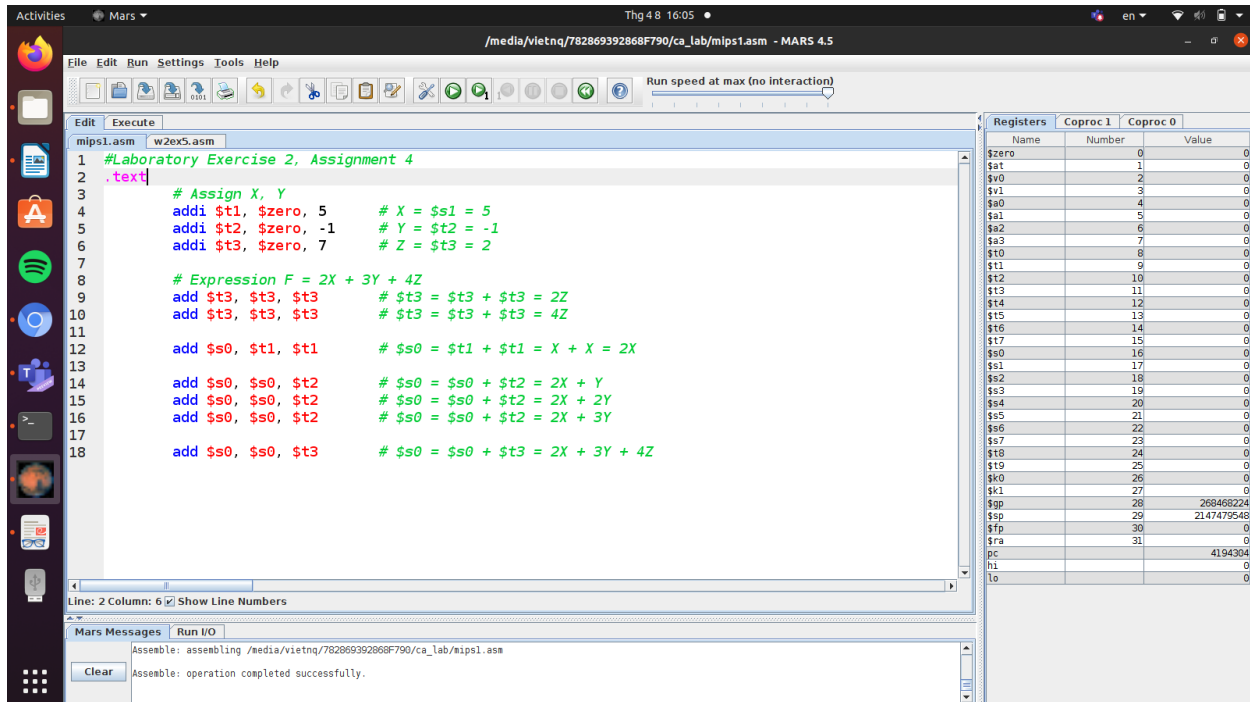
lui \$s0, 0x2110 => 0x00400000
ori \$s0, \$s0, 0x003d => 0x00400004

Assignment 3: *li* pseudo instruction

The value to be loaded by **li** is **0x2110003d** (*in 32-bit range*), while the operand is limited in range of 16 bits for the I-type instruction, so MIPS implemented 2 basic instructions for each half of the value instead of 1 instruction compared to the smaller value (**0x2**).

The ways these 2 basic instructions implemented are similar to the first 2 instruction in the case of **Assignment 1**.

Assignment 4: Calculating simple math expression $2X+Y$



```
1 #Laboratory Exercise 2, Assignment 4
2 .text
3
4 # Assign X, Y
5 addi $t1, $zero, 5    # X = $s1 = 5
6 addi $t2, $zero, -1   # Y = $t2 = -1
7 addi $t3, $zero, 7    # Z = $t3 = 2
8
9 # Expression F = 2X + 3Y + 4Z
10 add $t3, $t3, $t3     # $t3 = $t3 + $t3 = 2Z
11 add $t3, $t3, $t3     # $t3 = $t3 + $t3 = 4Z
12
13 add $s0, $t1, $t1     # $s0 = $t1 + $t1 = X + X = 2X
14
15 add $s0, $s0, $t2     # $s0 = $s0 + $t2 = 2X + Y
16 add $s0, $s0, $t2     # $s0 = $s0 + $t2 = 2X + 2Y
17 add $s0, $s0, $t2     # $s0 = $s0 + $t2 = 2X + 3Y
18 add $s0, $s0, $t3     # $s0 = $s0 + $t3 = 2X + 3Y + 4Z
```

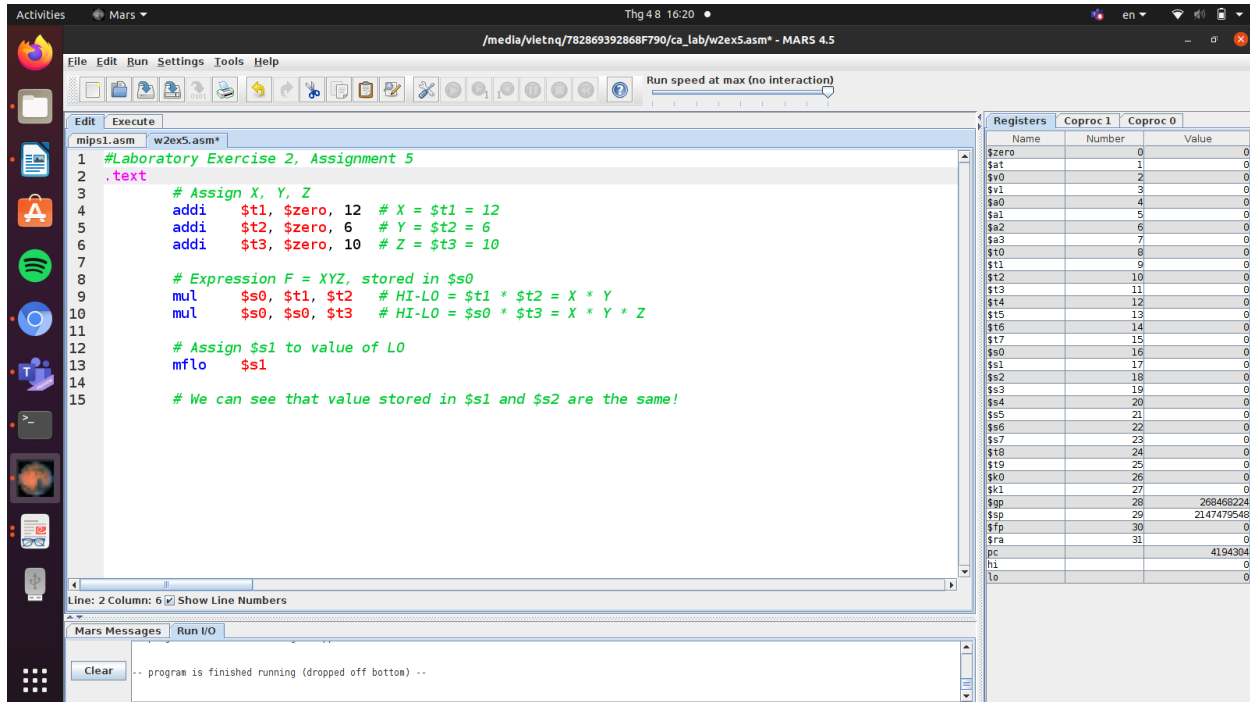
Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268460224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194364
hi		0
lo		0

Mars Messages Run I/O

Assemble: assembling /media/vietnq/782869392868F790/ca_lab/mips1.asm

Assemble: operation completed successfully.

Assignment 5: Multiplication



Assignment 6: Declaring and accessing variables

- The **la** instruction is implemented by 2 basic instructions **lui** and **ori**. For example, the machine code of these 2 instructions for variable X are: **0x3c011001** and **0x34380000**, where the last 4 numbers are 2 parts of the address of variable X (**10010000**).
- **lw**: load value of a variable in memory by its address, then store in the register.
- **sw**: assign a value for a variable in memory.