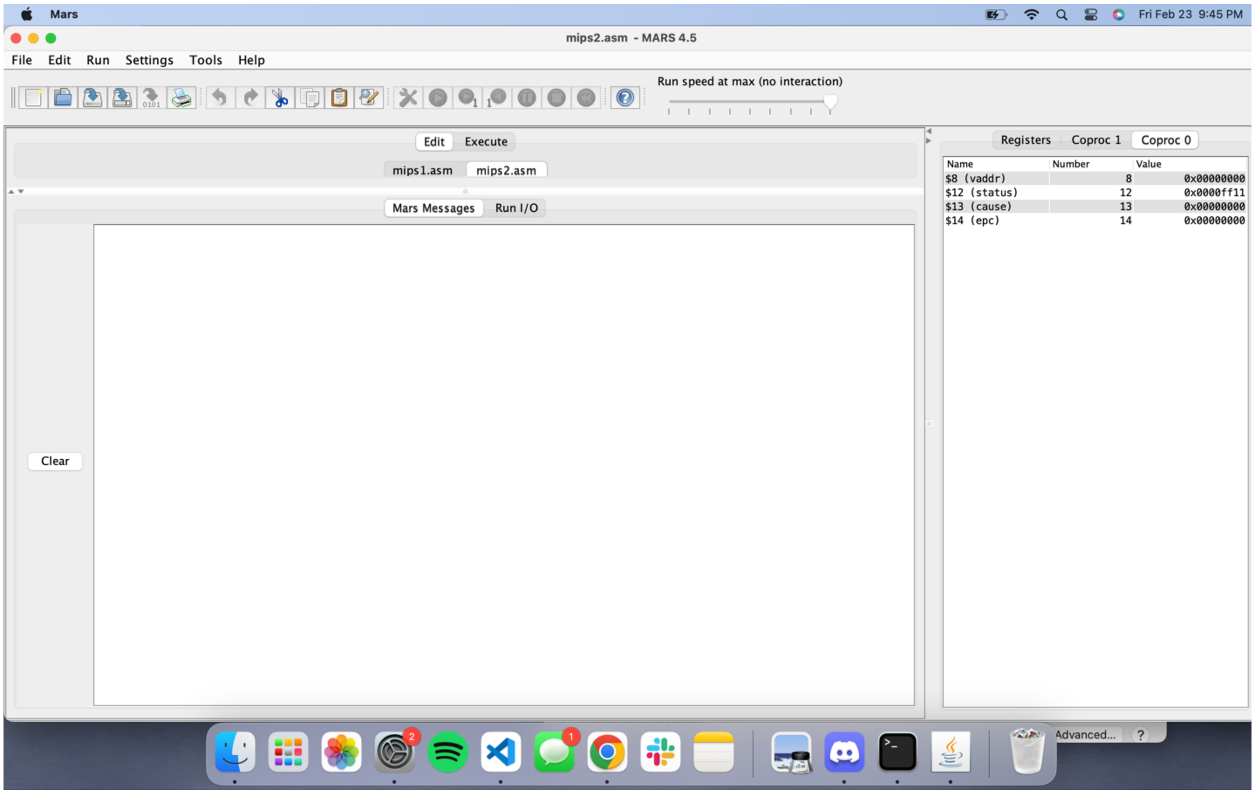
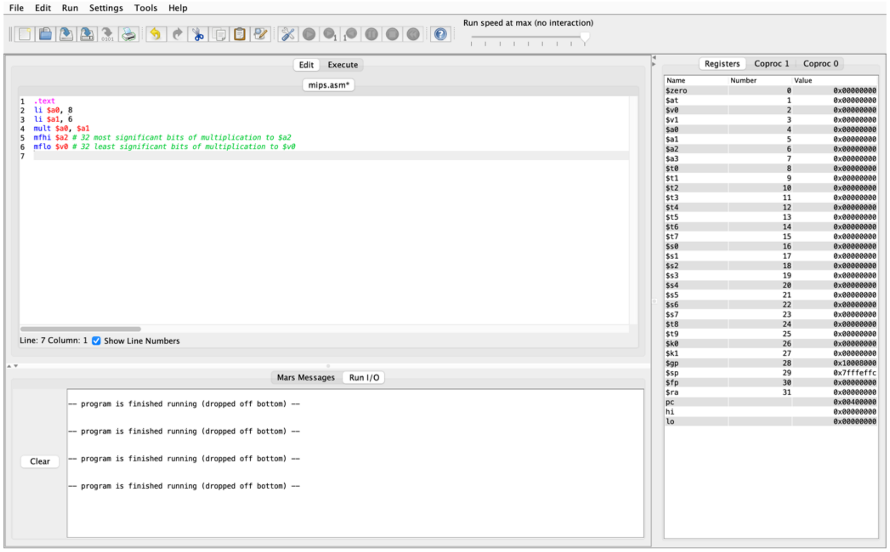
Clay Ramey – COMP 3350 – Lab 01 – Dr. Tu

1. SS from opening MARS after installing



Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff11
$13 (cause)	13	0x00000000
$14 (epc)	14	0x00000000

2-a. The changed code

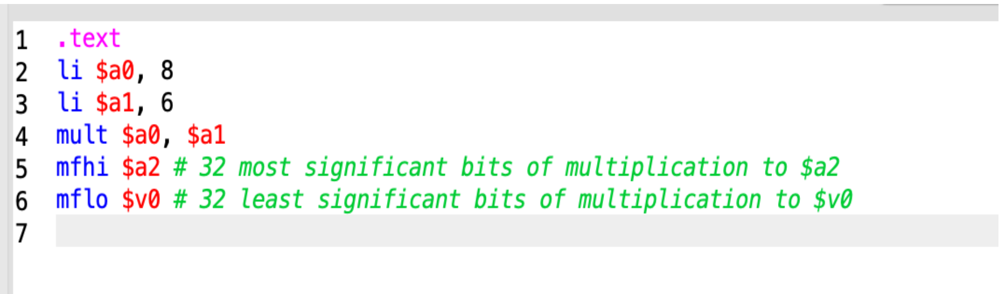

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
mips.asm*
Line: 7 Column: 1
☑ Show Line Numbers
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0

```
.text
li $a0, 8
li $a1, 6
mult $a0, $a1
mfhi $a2 # 32 most significant bits of multiplication to $a2
mflo $v0 # 32 least significant bits of multiplication to $v0

```

```
-- program is finished running (dropped off bottom) --

-- program is finished running (dropped off bottom) --

-- program is finished running (dropped off bottom) --

-- program is finished running (dropped off bottom) --
```

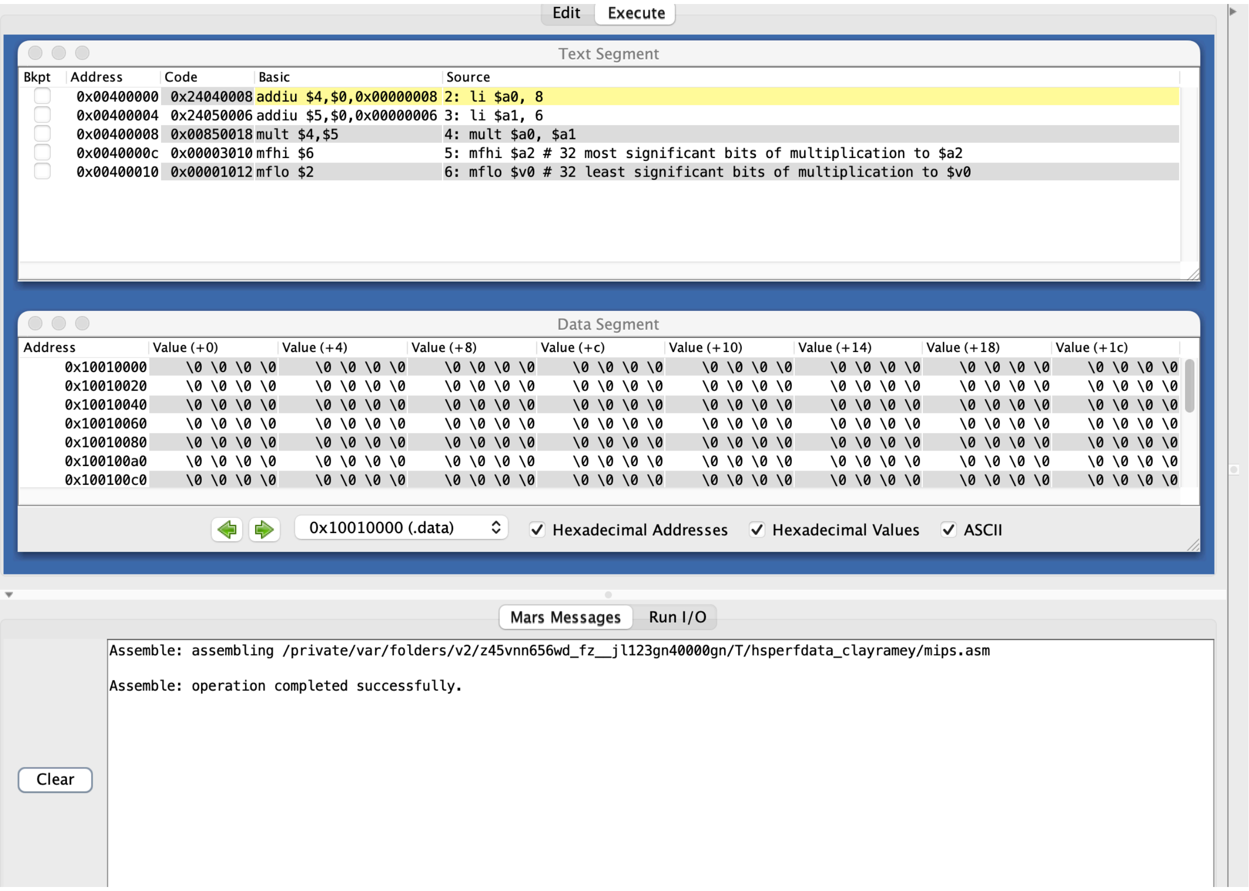
Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

2-a. more visible SS of the code



```
.text
li $a0, 8
li $a1, 6
mult $a0, $a1
mfhi $a2 # 32 most significant bits of multiplication to $a2
mflo $v0 # 32 least significant bits of multiplication to $v0
```

2-b. SS showing the code compiled below



Bkpt	Address	Code	Basic	Source
□	0x00400000	0x24040008	addiu $4,$0,0x00000008	2: li $a0, 8
□	0x00400004	0x24050006	addiu $5,$0,0x00000006	3: li $a1, 6
□	0x00400008	0x00850018	mult $4,$5	4: mult $a0, $a1
□	0x0040000c	0x00003010	mfhi $6	5: mfhi $a2 # 32 most significant bits of multiplication to $a2
□	0x00400010	0x00001012	mflo $2	6: mflo $v0 # 32 least significant bits of multiplication to $v0

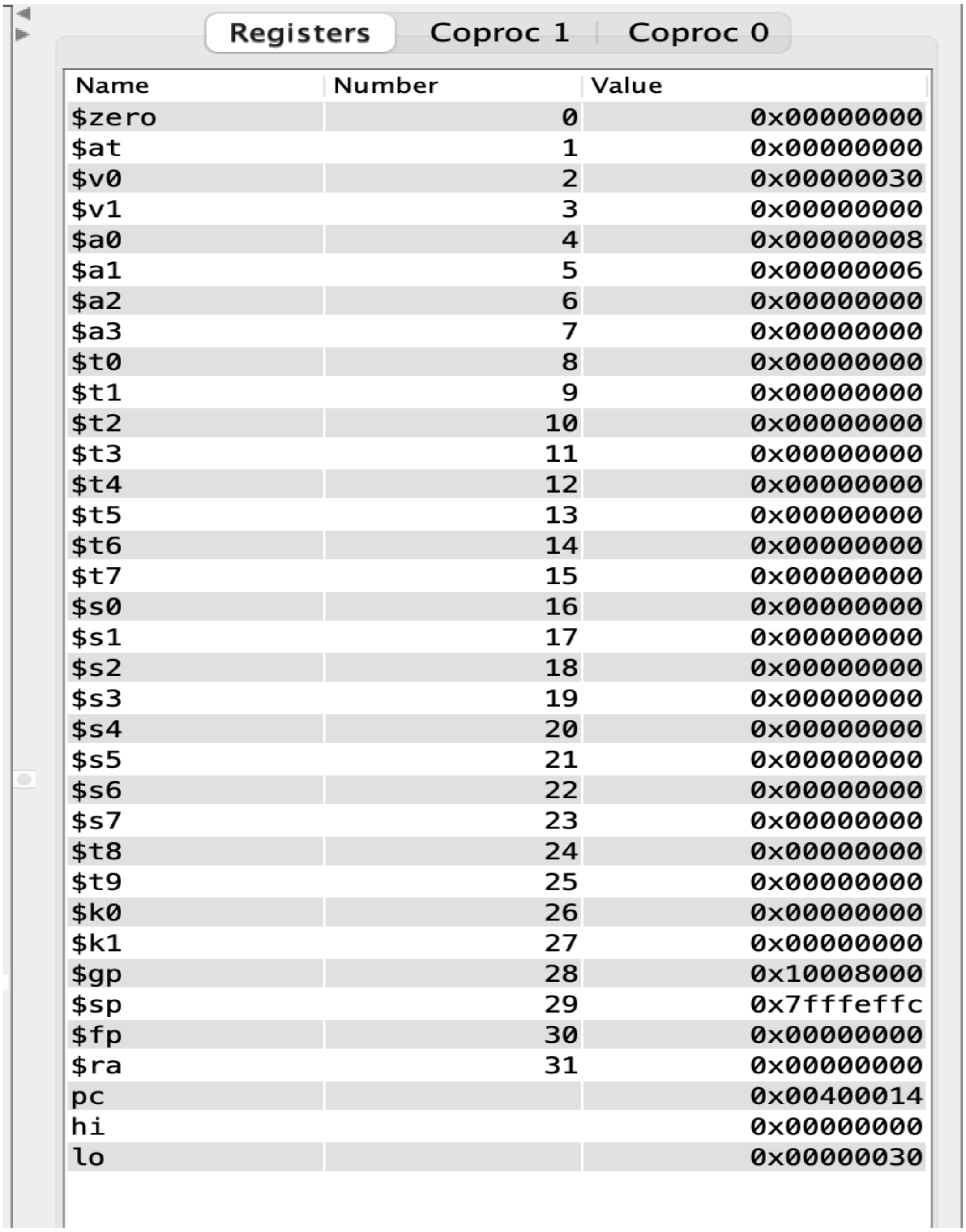
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values ☑ ASCII

```
Assemble: assembling /private/var/folders/v2/z45vnn656wd_fz__jl123gn40000gn/T/hsperfdata_clayramey/mips.asm

Assemble: operation completed successfully.
```

2-b. SS of registers after running previous code with changed values. Does the following:

1. Stores value ‘8’ into $a0
2. Stores value ‘6’ into $a1
3. Multiplies values in $a0 * $a1
4. Stores the values in $a2, then $v0, indication in step 5 & 6
5. Puts 32 MSB into $a2 (value shown below)
6. Puts 32 LSB into $v0 (value shown below)



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000030
$v1	3	0x00000000
$a0	4	0x00000008
$a1	5	0x00000006
$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
$s0	16	0x00000000
$s1	17	0x00000000
$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	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x00000030

Step 3 (bonus)

Case1: (yes – an exception occurred), because on step 6 of the code, the add command, it cut to coproc 0 – status changed in $12 to ff13 when failing (no output) (error happened because of overflow)

Case2: (no – an exception didn't occur), the code made it through every step and output:
-2147483617 | $12 (status) in coproc 0 ended as val = ff11

Case3: (no – an exception didn't occur), the code made it through every step and output:
-2147483648 | $12 (status) in coproc 0 ended as val = ff11

Case4: (yes – an exception occurred), because on step 6 of the code, the add command, it cut to coproc 0 – status went from val = ff11 in $12 to val = ff13 when failing (no output) (error happened because of overflow)

Case5: (no – an exception didn't occur), the code made it through every step and output:
-2 | $12 (status) in coproc 0 ended as val = ff11