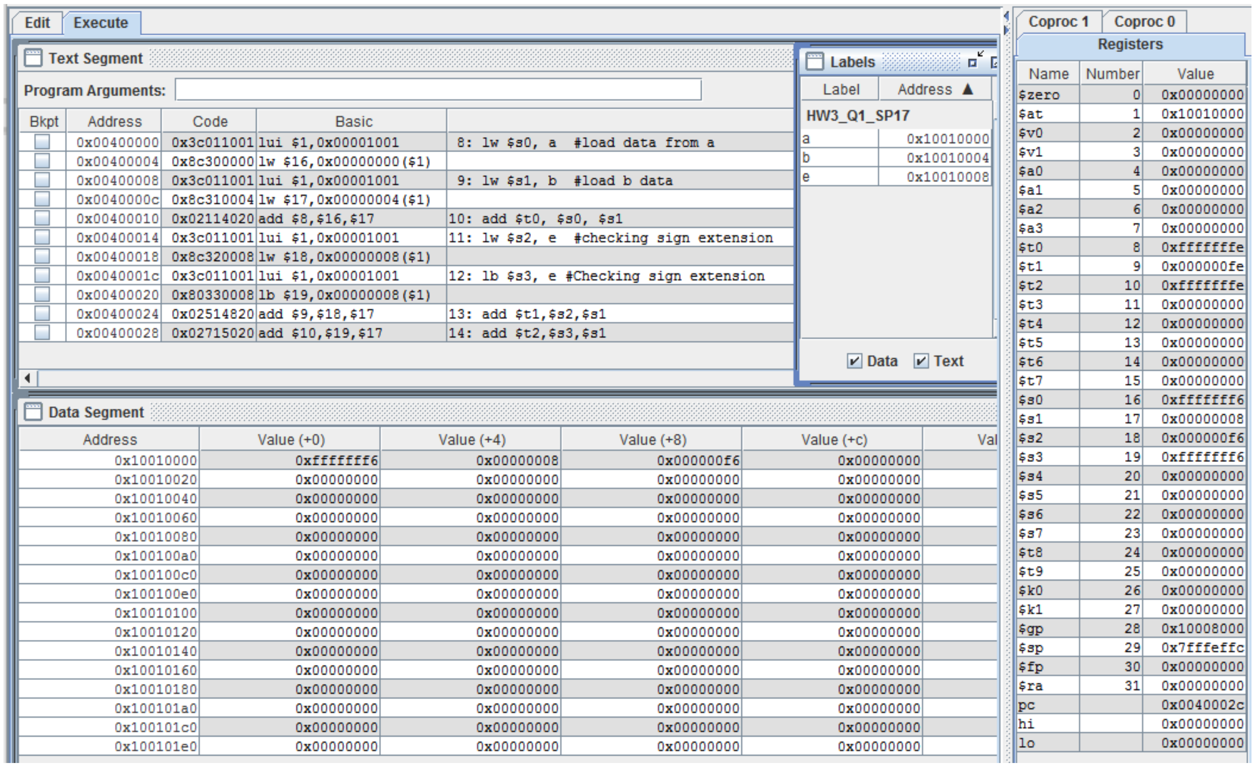
HW#4 (CSC390-Spring 2018)
Due: 02/16/2018 by 11:00PM (Blackboard Submission)

Q1. Consider the following MIPS assembly codes which perform a simple mathematical operation (-10+8) using the variables a, b, and e. The value -10 is assigned to variables a and e, as shown in lines 2 and 4, respectively.

```
1   .data    #declare data segment
2   a: .word -10
3   b: .word 8
4   e: .byte -10
5
6   .text #code segment
7
8   lw $s0, a  #load data from a
9   lw $s1, b  #load b data
10  add $t0, $s0, $s1
11  lw $s2, e  #checking sign extension
12  lb $s3, e #Checking sign extension
13  add $t1,$s2,$s1
14  add $t2,$s3,$s1
```

When we assembled the program, the first three locations of the Data-Segment are initialized with the values of a, b, and e, as shown in the following figure. Explain the questions on the next page.



Text Segment

Program Arguments:

Bkpt	Address	Code	Basic	
□	0x00400000	0x3c011001	lui $1,0x00001001	8: lw $s0, a #load data from a
□	0x00400004	0x8c300000	lw $16,0x00000000($1)	
□	0x00400008	0x3c011001	lui $1,0x00001001	9: lw $s1, b #load b data
□	0x0040000c	0x8c310004	lw $17,0x00000004($1)	
□	0x00400010	0x02114020	add $8,$16,$17	10: add $t0, $s0, $s1
□	0x00400014	0x3c011001	lui $1,0x00001001	11: lw $s2, e #checking sign extension
□	0x00400018	0x8c320008	lw $18,0x00000008($1)	
□	0x0040001c	0x3c011001	lui $1,0x00001001	12: lb $s3, e #Checking sign extension
□	0x00400020	0x80330008	lb $19,0x00000008($1)	
□	0x00400024	0x02514820	add $9,$18,$17	13: add $t1,$s2,$s1
□	0x00400028	0x02715020	add $10,$19,$17	14: add $t2,$s3,$s1

Labels

Label	Address ▲
HW3_Q1_SP17	
a	0x10010000
b	0x10010004
e	0x10010008

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Val
0x10010000	0xfffffff6	0x00000008	0x000000f6	0x00000000	
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	

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	0xfffffffe
$t1	9	0x000000fe
$t2	10	0xfffffffe
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0xfffffff6
$s1	17	0x00000008
$s2	18	0x000000f6
$s3	19	0xfffffff6
$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		0x0040002c
hi		0x00000000
lo		0x00000000

i) Observe that for the same value of -10 the first location of the Data-Segment has 0xfffffff6 and the third location has 0x000000f6. Explain why?

ii) Line 10 of the code is performing (-10+8) operation. Check whether you are getting the correct result in $t0.

iii) Lines 11 and 12 are loading the value of the variable “e” into the registers $s2 and $s3 respectively. Observe that after executing the program $s2 has 0x000000f6 and $s3 has 0xfffffff6. Explain Why?

iv) Observe that lines 13 and 14 are performing the same mathematical operation, (-10+8), and storing the results in $t1 and $t2, respectively. Check which one has the correct result. Explain why?

Q2. Write a MIPS assembly code to transfer data from register $s0 to $t0 without using Load and Store instructions.

Q3. Write down the **machine code** of the following R-format instruction showing every instruction fields.

add $t3, $S3, $S4

Verify your machine code using the MARS simulator.

Q4. Write a MIPS assembly language program that calls a procedure, Add_Sub_Mul, which accept four parameters (g,h,i,j) and returns,

f = (g+h) if i >j; f= (g-h) if i < j; and f = g*h if i == j; the equivalent C function is shown below:

```
int Add_Sum_Mul (int g, int h, int i, int j) {
  int f;
 if  (i > j) {
          f=(g+h);}
  else if (i < j) {
          f=(g-h);}
 else if (i==j){
        f=g*h}
       }
```

Consider the variables g, h, i, j and f are initialized with some initial values in the data segment. Use $s0 as f in the function and also use $s0 to store the base address of f in the memory location. Clearly comment on the every instruction you use in your program. Specially, clearly show and describe the stack operation. Remember, resisters ($a0-$a2) are used for passing arguments in to the function and $v resisters are used to store the results in the function.