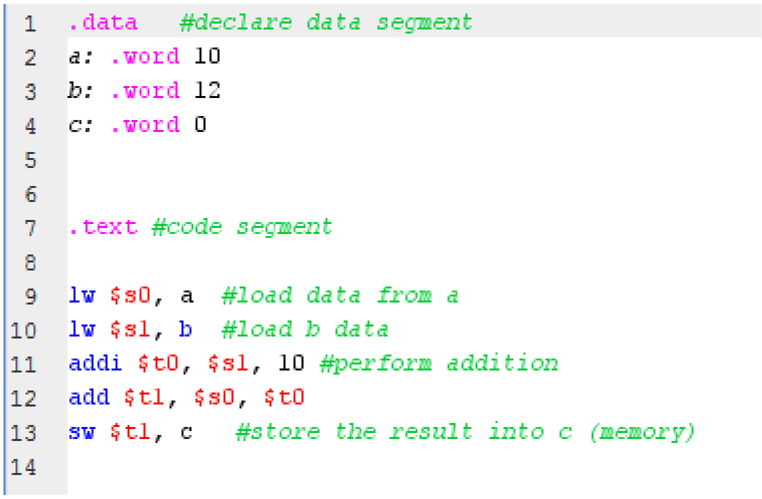
HW#3 (CSC390-Sp2018)-Solution

Q1.

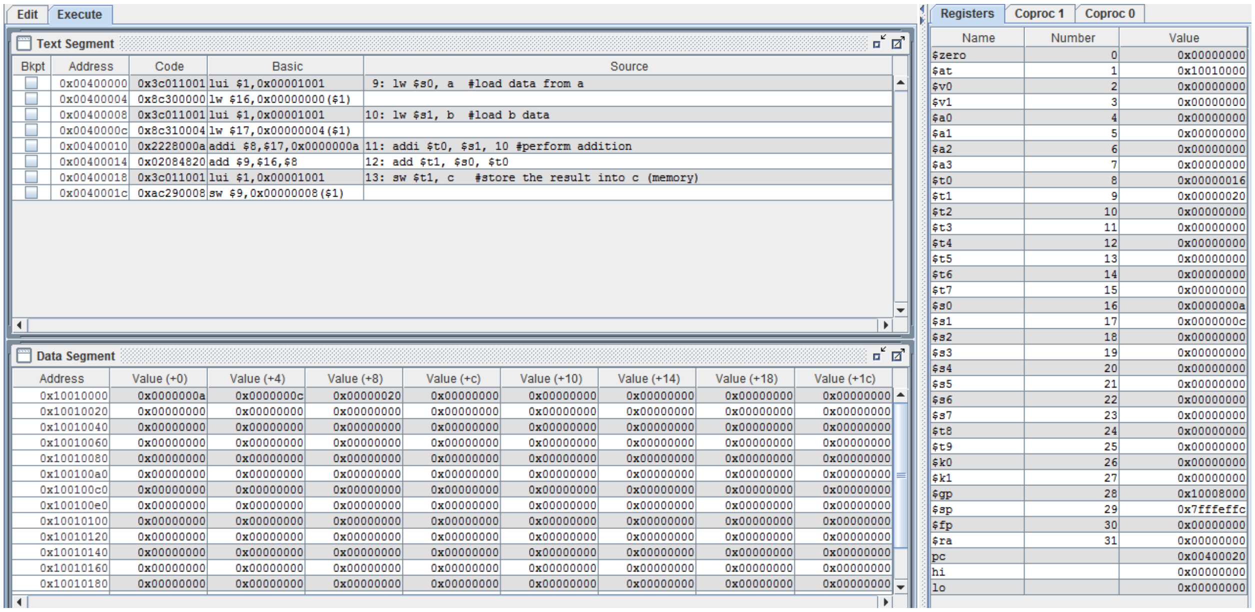


```
.data   #declare data segment
a: .word 10
b: .word 12
c: .word 0


.text #code segment

lw $s0, a  #load data from a
lw $s1, b  #load b data
addi $t0, $s1, 10 #perform addition
add $t1, $s0, $t0
sw $t1, c   #store the result into c (memory)

```



Text Segment

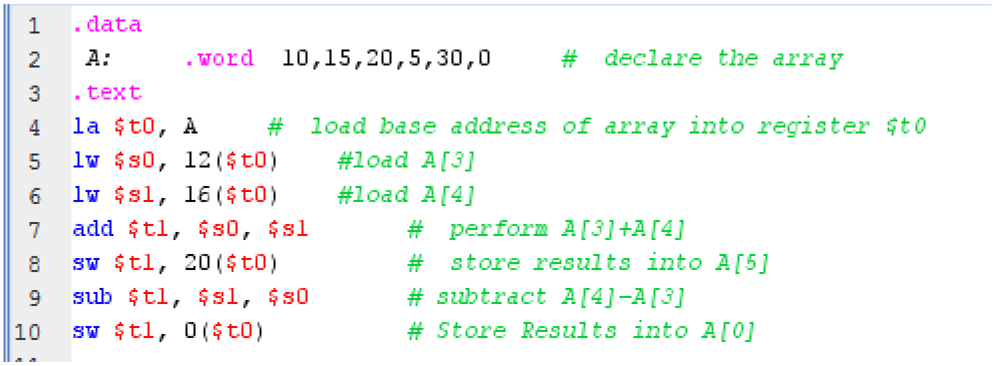
Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x3c011001	lui $1,0x00001001	9: lw $s0, a #load data from a
☐	0x00400004	0x8c300000	lw $16,0x00000000($1)	
☐	0x00400008	0x3c011001	lui $1,0x00001001	10: lw $s1, b #load b data
☐	0x0040000c	0x8c310004	lw $17,0x00000004($1)	
☐	0x00400010	0x2228000a	addi $8,$17,0x0000000a	11: addi $t0, $s1, 10 #perform addition
☐	0x00400014	0x02084820	add $9,$16,$8	12: add $t1, $s0, $t0
☐	0x00400018	0x3c011001	lui $1,0x00001001	13: sw $t1, c #store the result into c (memory)
☐	0x0040001c	0xac290008	sw $9,0x00000008($1)	

Data Segment

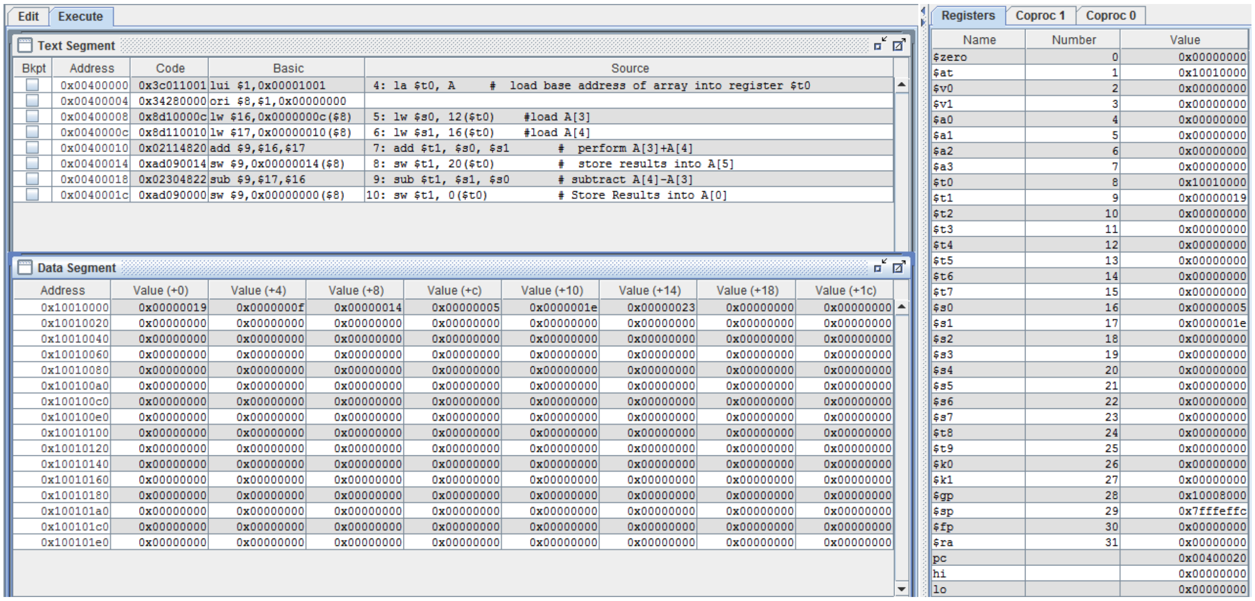
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000a	0x0000000c	0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	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	0x00000016
$t1	9	0x00000020
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x0000000a
$s1	17	0x0000000c
$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		0x00400020
hi		0x00000000
lo		0x00000000

Q2.



```
.data
 A:      .word  10,15,20,5,30,0      #  declare the array
.text
la $t0, A      #  load base address of array into register $t0
lw $s0, 12($t0)     #load A[3]
lw $s1, 16($t0)     #load A[4]
add $t1, $s0, $s1        #  perform A[3]+A[4]
sw $t1, 20($t0)          #  store results into A[5]
sub $t1, $s1, $s0        # subtract A[4]-A[3]
sw $t1, 0($t0)           # Store Results into A[0]
```



Text Segment

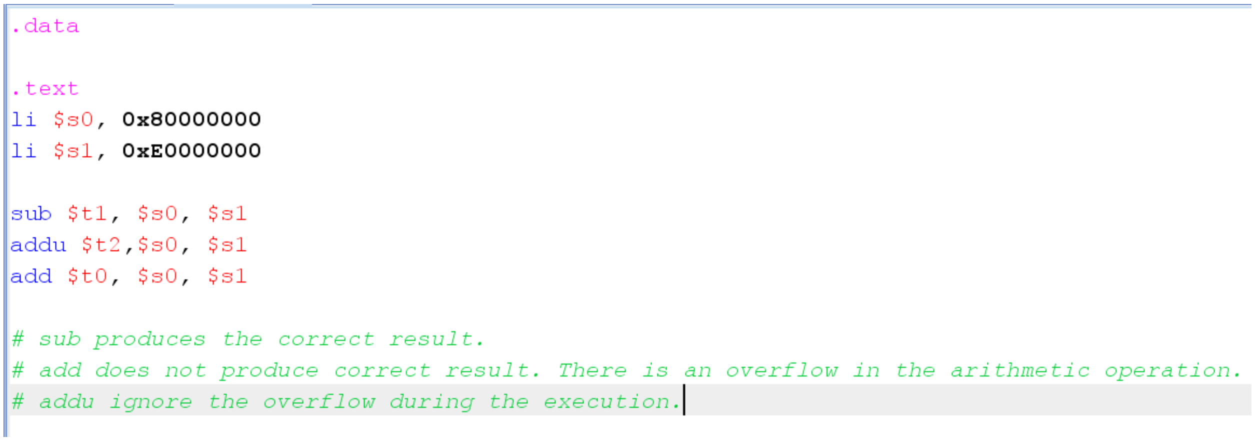
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c011001	lui $1,0x00001001	4: la $t0, A # load base address of array into register $t0
□	0x00400004	0x34280000	ori $8,$1,0x00000000	
□	0x00400008	0x8d10000c	lw $16,0x0000000c($8)	5: lw $s0, 12($t0) #load A[3]
□	0x0040000c	0x8d110010	lw $17,0x00000010($8)	6: lw $s1, 16($t0) #load A[4]
□	0x00400010	0x02114820	add $9,$16,$17	7: add $t1, $s0, $s1 # perform A[3]+A[4]
□	0x00400014	0xad090014	sw $9,0x00000014($8)	8: sw $t1, 20($t0) # store results into A[5]
□	0x00400018	0x02304822	sub $9,$17,$16	9: sub $t1, $s1, $s0 # subtract A[4]-A[3]
□	0x0040001c	0xad090000	sw $9,0x00000000($8)	10: sw $t1, 0($t0) # Store Results into A[0]

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000019	0x0000000f	0x00000014	0x00000005	0x0000001e	0x00000023	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	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	0x10010000
$t1	9	0x00000019
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000005
$s1	17	0x0000001e
$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		0x00400020
hi		0x00000000
lo		0x00000000

Q3.

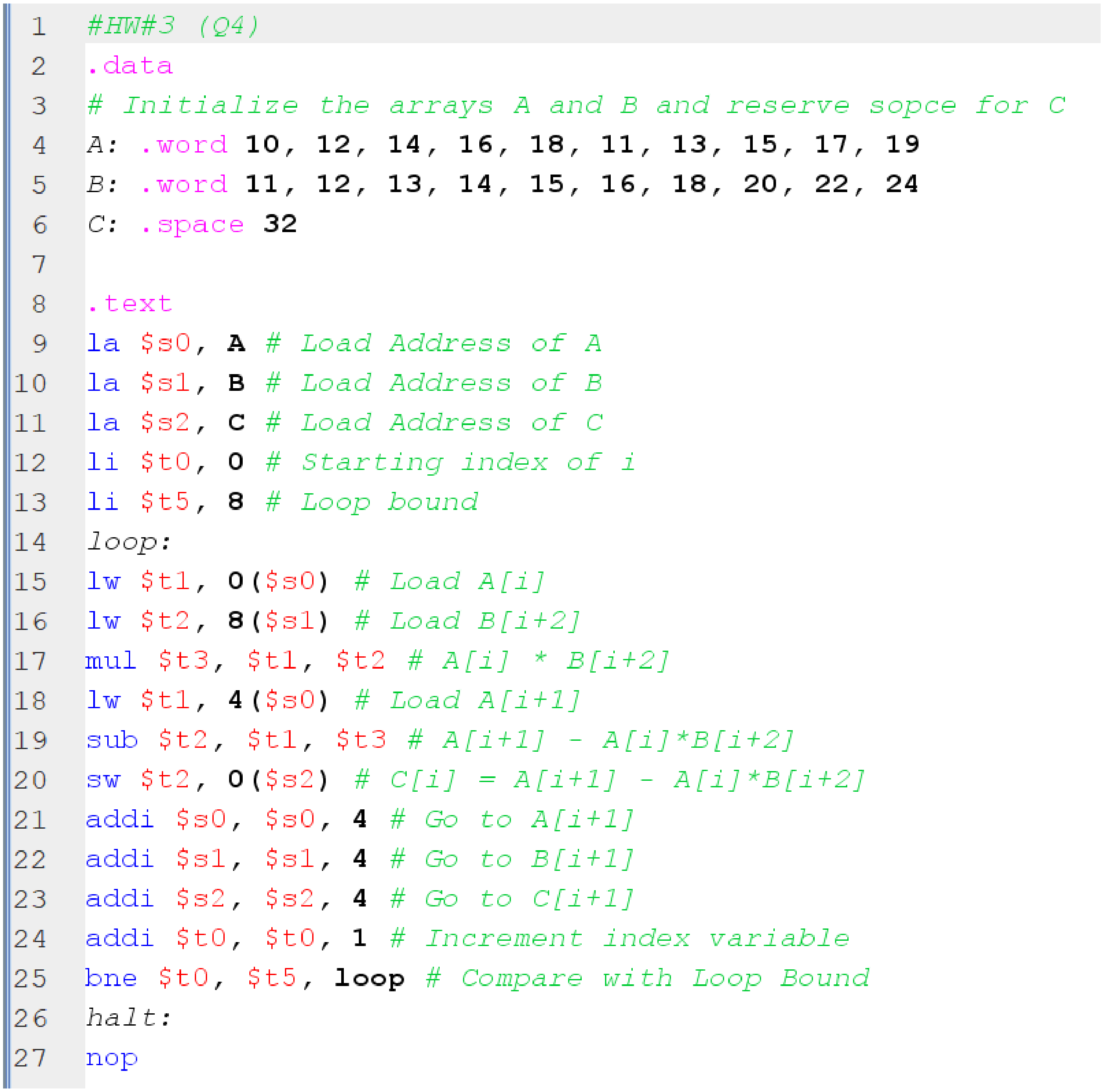


```
.data

.text
li $s0, 0x80000000
li $s1, 0xE0000000

sub $t1, $s0, $s1
addu $t2,$s0, $s1
add $t0, $s0, $s1

# sub produces the correct result.
# add does not produce correct result. There is an overflow in the arithmetic operation.
# addu ignore the overflow during the execution.
```



```
#HW#3 (Q4)
.data
# Initialize the arrays A and B and reserve sopce for C
A: .word 10, 12, 14, 16, 18, 11, 13, 15, 17, 19
B: .word 11, 12, 13, 14, 15, 16, 18, 20, 22, 24
C: .space 32

.text
la $s0, A # Load Address of A
la $s1, B # Load Address of B
la $s2, C # Load Address of C
li $t0, 0 # Starting index of i
li $t5, 8 # Loop bound
loop:
lw $t1, 0($s0) # Load A[i]
lw $t2, 8($s1) # Load B[i+2]
mul $t3, $t1, $t2 # A[i] * B[i+2]
lw $t1, 4($s0) # Load A[i+1]
sub $t2, $t1, $t3 # A[i+1] - A[i]*B[i+2]
sw $t2, 0($s2) # C[i] = A[i+1] - A[i]*B[i+2]
addi $s0, $s0, 4 # Go to A[i+1]
addi $s1, $s1, 4 # Go to B[i+1]
addi $s2, $s2, 4 # Go to C[i+1]
addi $t0, $t0, 1 # Increment index variable
bne $t0, $t5, loop # Compare with Loop Bound
halt:
nop
```