ECE 346 HW # 1

Bradley McKee

UIN: 00975338

Question 1)

lui $s0, 0x1001

op rs rt imm

001111 00000 10000 0001000000000001

xori $s1, $s0, -10

op rs rt imm

001110 10000 10001 1111111111110111

lui $s2,0x222

op rs rt imm

001111 00000 10010 0000001000100010

add $s3,$s1,$s2

op rs rt rd sham funct

000000 10001 10010 10011 00000 100000

sw $s3,4($s0)

op rs rt im

101011 10000 10011 0000000000000100

1.2)

Function Type Register address affected

lui $s0 0x10010000

xori $s1 0x1000FFF7

lui $s2 0x02220000

add $s3 0x1222FFF7

sw all take contents of $s3 offset it by 4 bits of the base address ($s0)

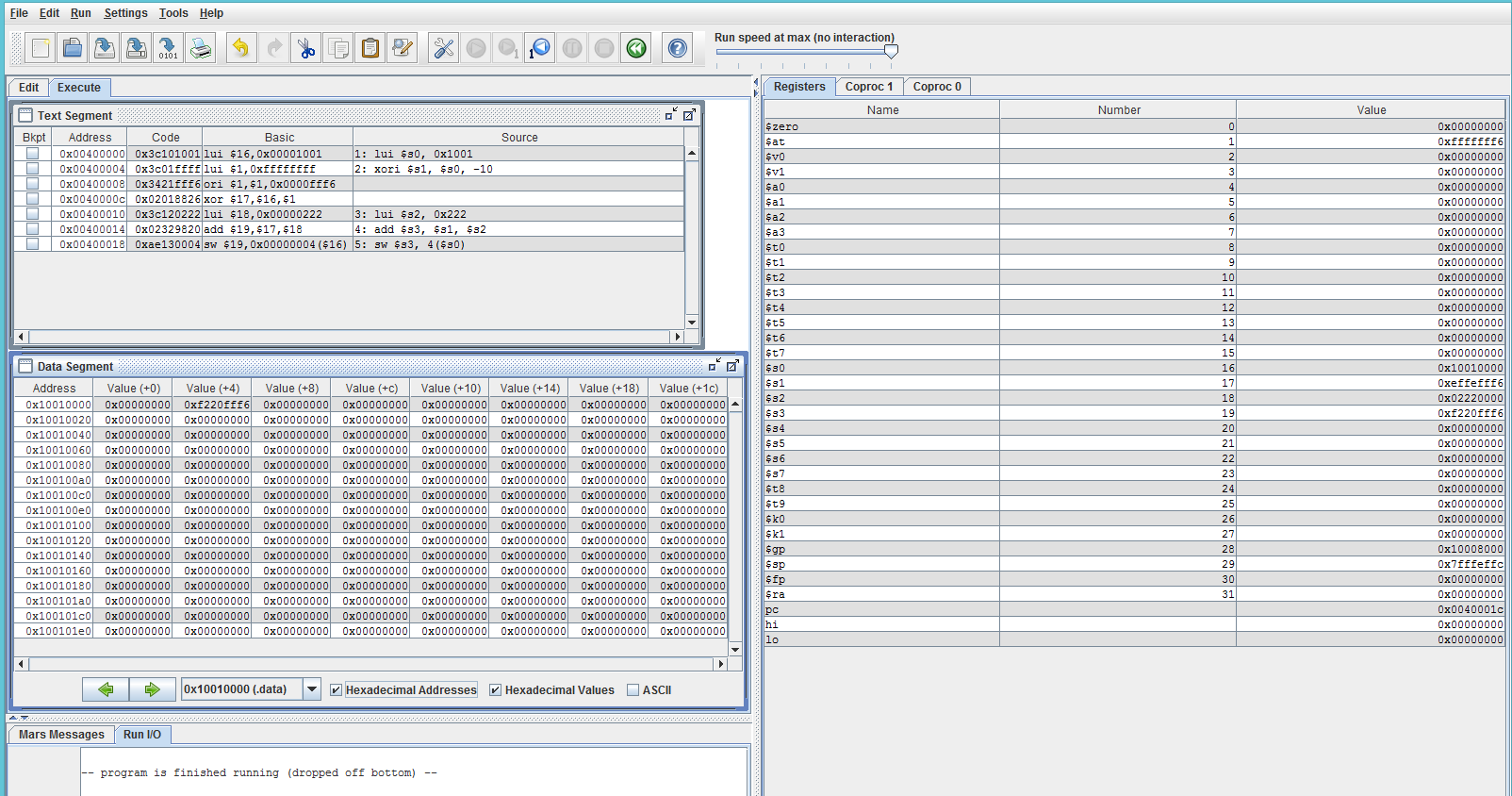
Final addresses:

$s0 0x00010000

$s1 0x1000FFF7

$s2 0x02220000

$s3 0x1222FFF7



1.3) The code doesn’t match from my handwork, from what I noticed it had to break up the xori into 2 commands

1.4) after the program was done, $at is equivalent to $s3 which makes me believe that the register $at was kind of used as a temporary when executing through the different instructions.

Question 2)

0x20080000

0010 00|00 000|0 1000| 0000 0000 0000 0000

op rs rt imm

addi $t0,0

0x20090001

0010 00|00 000|0 1001| 0000 0000 0000 0001

op rs rt imm

addi $t1,0x1

0x0089502A

0000 00|00 100|0 1001| 0101 0|000 00|01 1010

op rs rt rd sham func

SLT $t2,$a0,$t1

0X15400003

0001 01|01 010|0 0000| 0000 0000 0000 0011

op rs rt imm

BNE $t2, $0,0x3

0X01094020

0000 00|01 000|0 1001| 0100 0|000 00|10 0000

Op rs rt rd sham func

add $t0, $t0, $t1

0X21290002

0010 00|01 001|0 1001| 0000 0000 0000 0010

Op rs rt imm

addi $t1,$t1,0x2

0X08100002

0000 10|00 0001 0000 0000 0000 0000 00010

op jump target address

j 0x00400008

0X01001020

0000 00|01 000|0 0000| 0001 0|000 00|10 0000

op rs rt rd shamt func

add $v0, $t0,$0

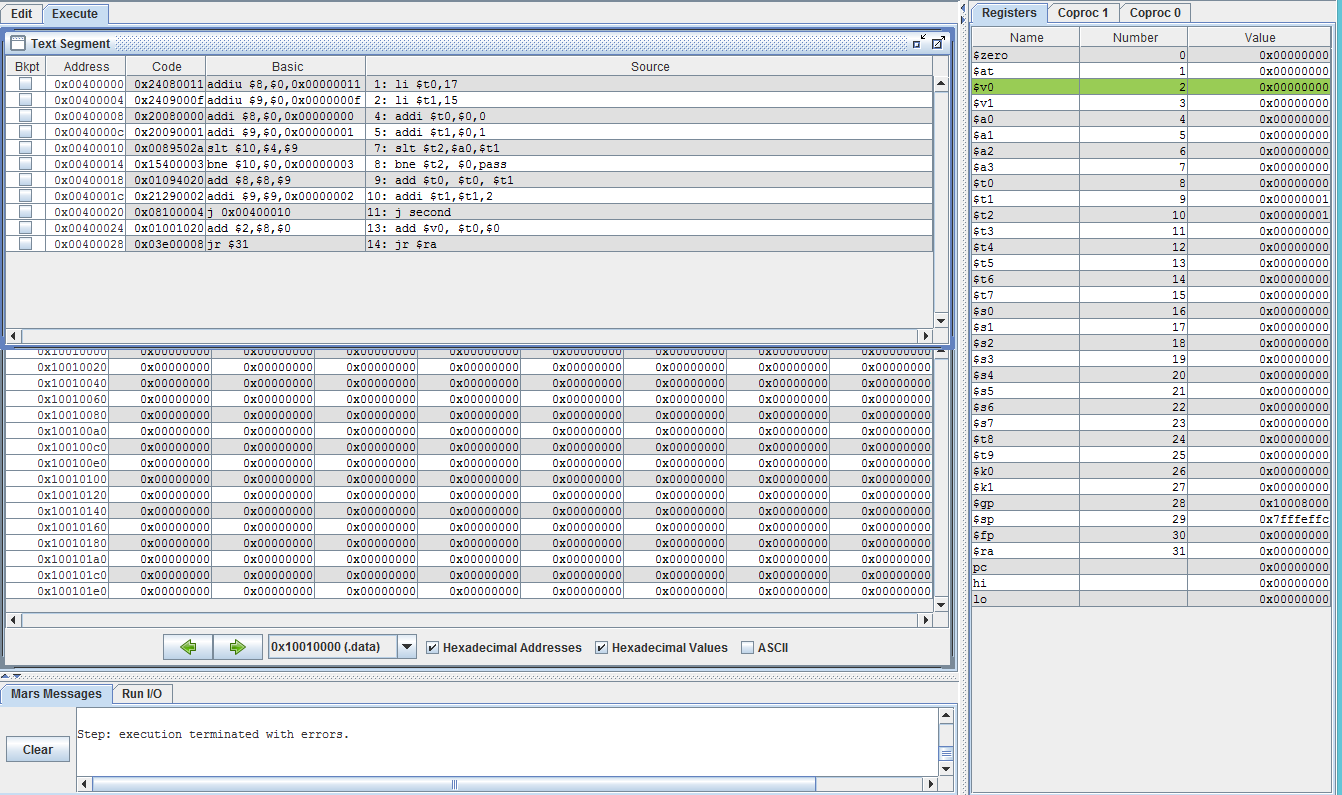
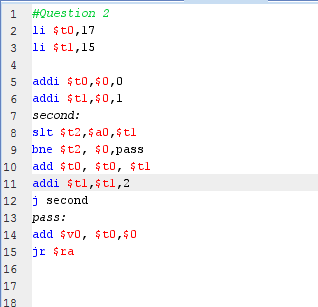
0X03E00008

0000 00|11 111|0 0000| 0000 0|000 00|00 1000

op rs r t rd shamt func

jr $ra

2.3)



The functionality of this program is to iterate through the values and when register $t2 is not equal to the value in register $0 (0), then it will pass to the adding line. Lastly, it will then jump to the register $ra then the program will terminate.

3a)

slt $t0,$s1,$s0

beq $t0,$0, ELSE

add $t1,$s0,$s1

J ENDPROGRAM

ELSE:

Sub $t1,$s0,$s1

ENDPROGRAM:

3b)

slt $t0,$s1,$s0

beq $s1,$s0, GLABEL

beq $t0,$0, HLABEL

GLABEL:

addi $s0,$s0,1

J ENDPROGRAM

HLABEL:

subi $s1,$s1,1

ENDPROGRAM:

3c)

slt $t0, $s0,$s1

beq $s0,$s1, GLABEL

beq $t0,$0, HLABEL

GLABEL:

Add $s0, $0, $0

J ENDPROGRAM

HLABEL:

add $s1,$0, $0

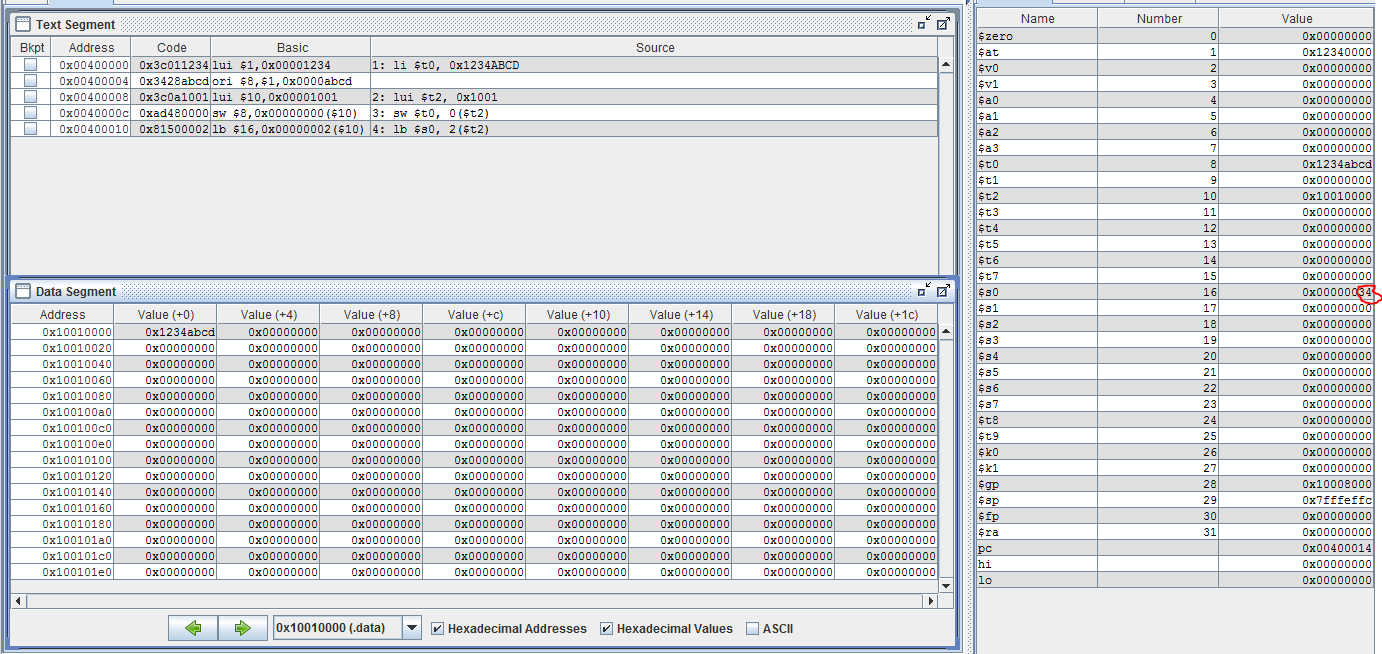
ENDPROGRAM:

Question 4)

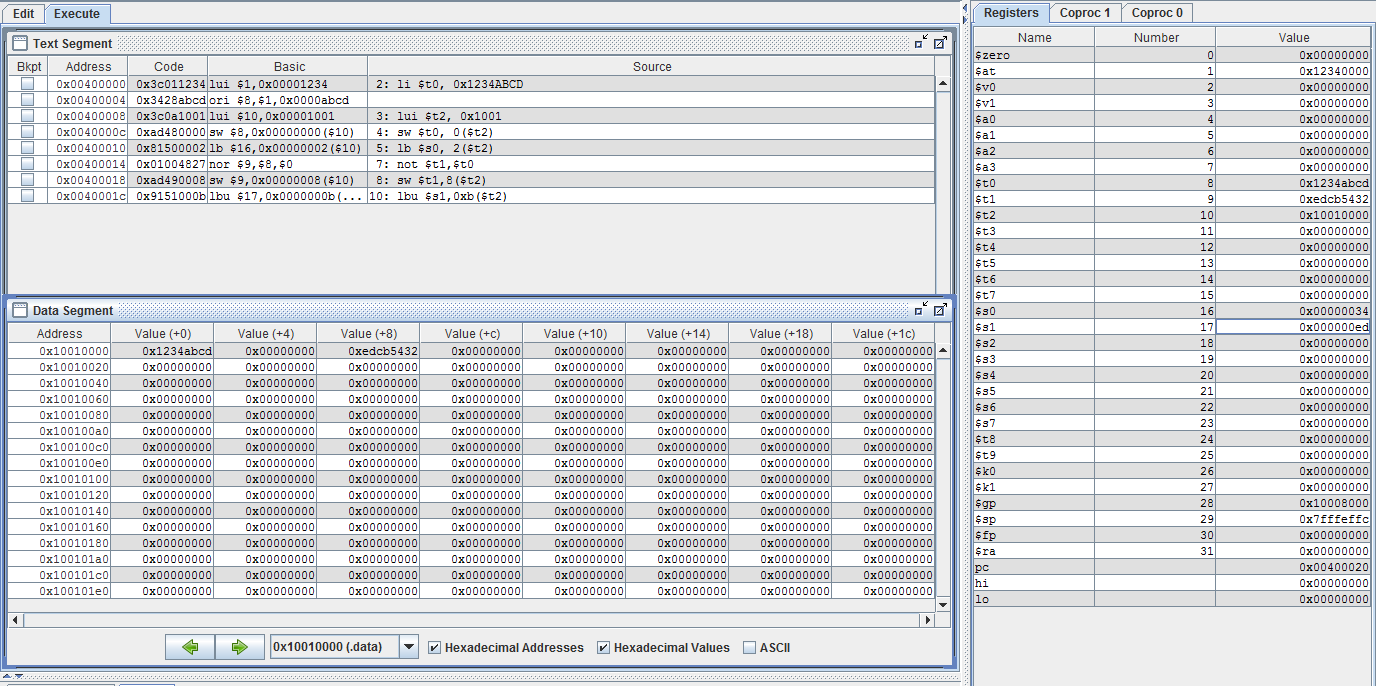
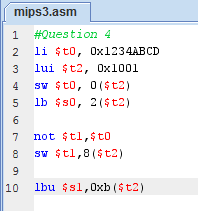
This will produce the output of the second byte from the stored address in $t2. Big-endian would be AB and little-endian would be 34.

4.2)

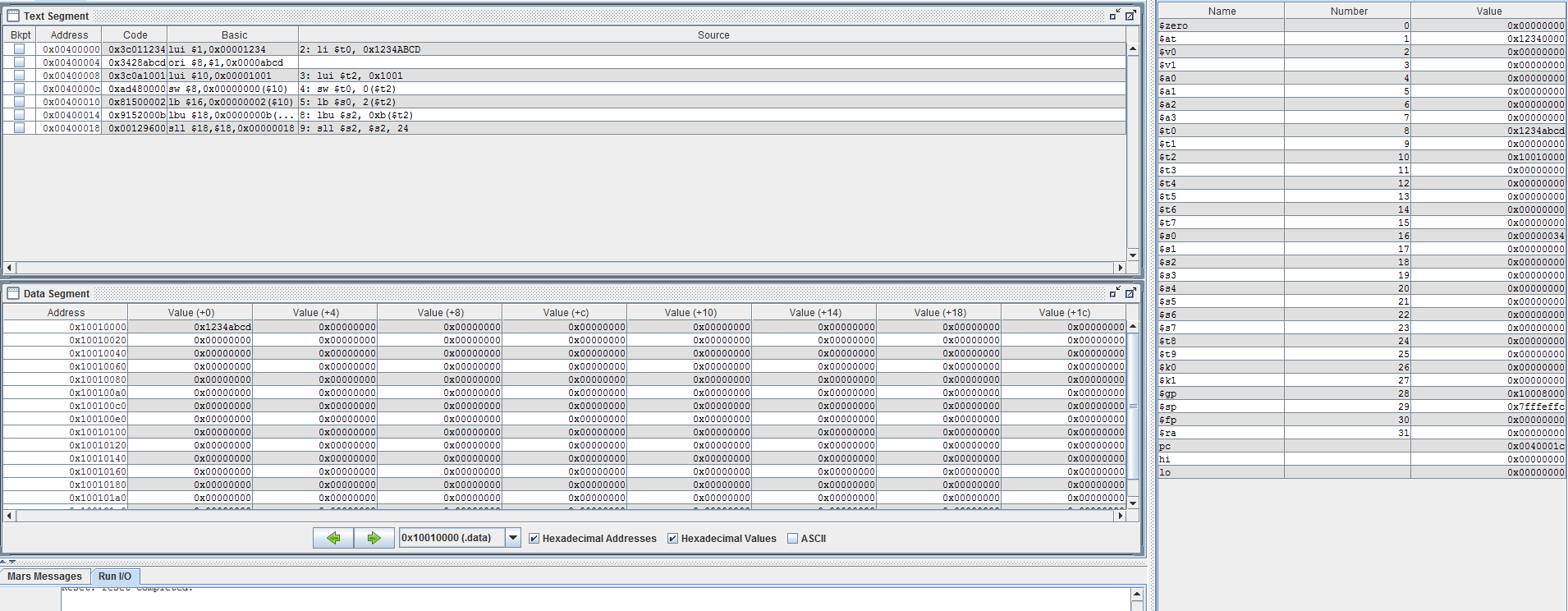
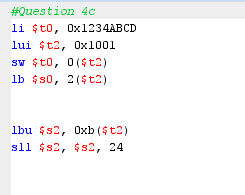
From the MIPS Simulation, it says that the second byte is 34, which would make MIPS a little-endian architecture.



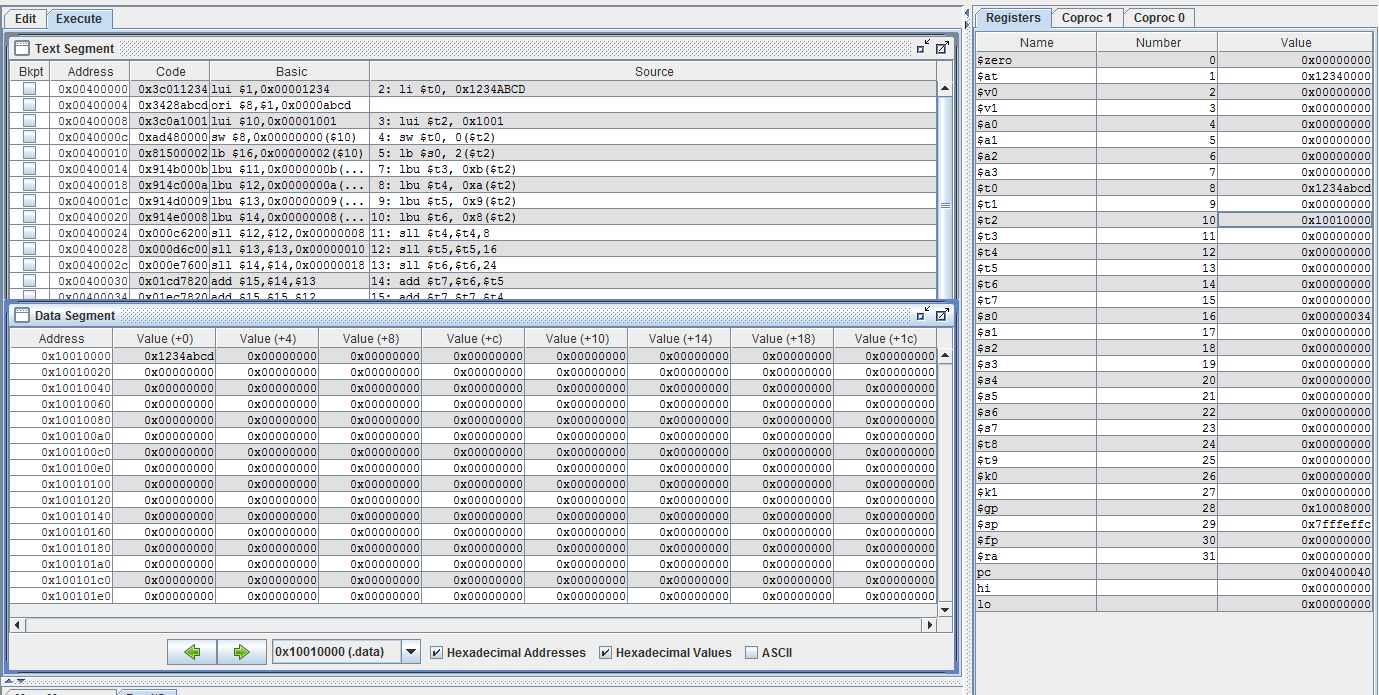
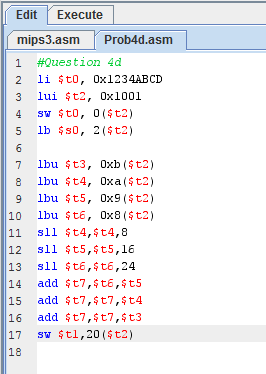
4.3a/b)



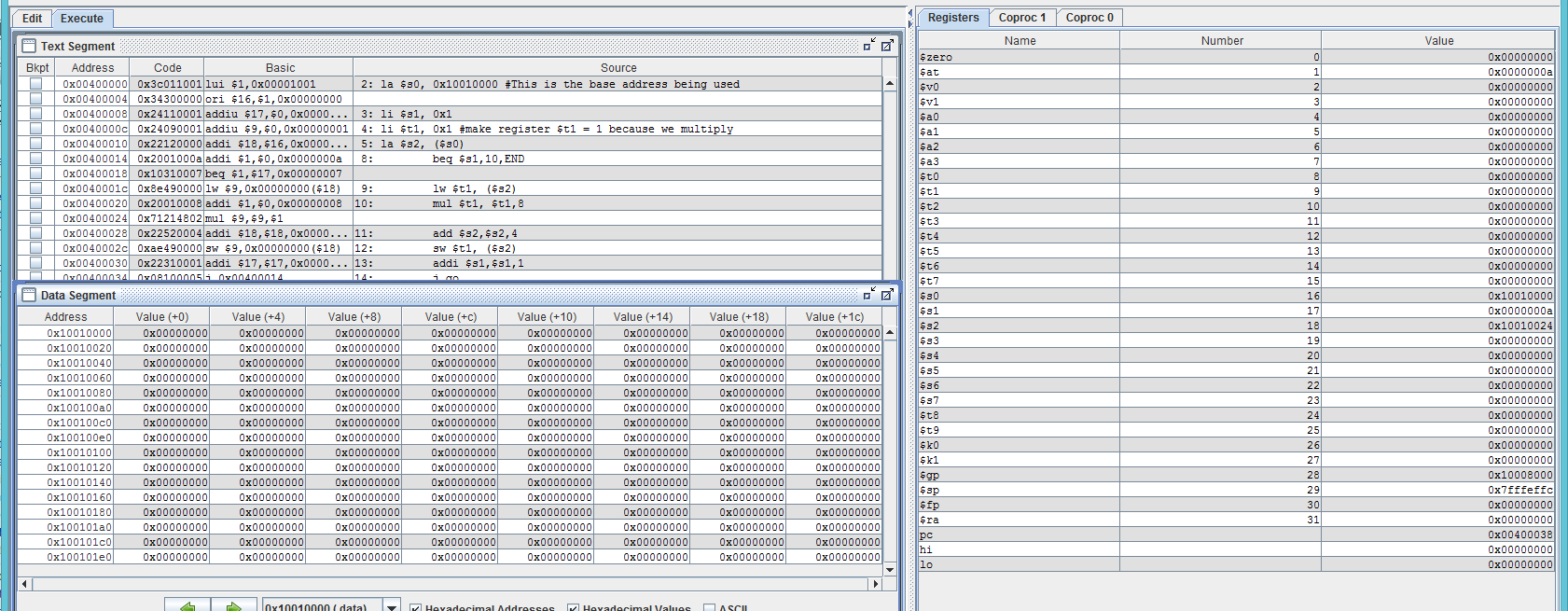
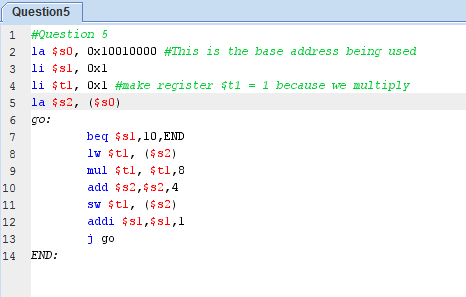
4.3c)



4.3d)



5)



6)

Regwrite: 1

RegDest: 1

ALUSrc: 01

Branch: 0

MemWrite: 0

MemtoReg: 0

ALUControl: 00 (whichever 2-bit binary makes the ALU add)