

#LabPart1.asm

#Z = (A-B) \* (C+D) + (E-F) - (A/C);

```
#####  
#                                     #  
#          DTAT SEGMENT              #  
#                                     #  
#####
```

.data

```
A: .word 15    # int A = 15  
B: .word 10    # int B = 10  
C: .word 5     # int C = 5  
D: .word 2     # int D = 2  
E: .word 18    # int E = 18  
F: .word -3    # int F = -3  
Z: .word 0     # int Z = 0
```

```
#####  
#                                     #  
#          TEXT SEGMENT              #  
#          MAIN CODE START NOW      #  
#                                     #  
#####
```

.text

main:

```
lw $t0, A      #load word $t0 = A = 15  
lw $t1, B      #load word $t1 = B = 10  
subu $t0, $t0, $t1  # $t0 = A-B = $t0 - $t1
```

```
lw $t1, C      # load word $t1 = C = 5  
lw $t2, D      # load word $t0 = D = 2  
addu $t1, $t1, $t2  # $t1 = C+D = $t1 + $t2
```

```
mult $t0, $t1   # (A-B) * (C+D) = $t0 * $t1
```

```
mflo $t0        # $t0 = lo = (A-B) * (C+D) = $t0 * $t1
```

```
lw $t1, E      # load word $t1 = E = 18
```

```
lw $t2, F      # load word $t1 = F = -3
subu $t1, $t1, $t2  # $t1 = E-F = $t1 - $t2
```

```
lw $t2, A      # load word $t2 = A = 15
lw $t3, C      # load word $t3 = C = 5
div $t2, $t3    # A/C = $t2 / $t3
```

```
mflo $t2       # $t2 = lo = A/C = $t2 / $t3
```

```
subu $t1, $t1, $t2  # $t1 = (E-F) - (A/C) = $t1 - $t2
```

```
addu $t0, $t0, $t1  # $t0 = (A-B) * (C+D) + (E-F) - (A/C) = $t0 + $t1
```

```
sw $t0, Z       # store word Z = $t0
```

```
lw $a0, Z       # load word $a0 = Z
li $v0, 1       # print the integer $a0
syscall
```

```
#####
#                                     #
#      EXIT FROM PROGRAM            #
#                                     #
#####
```

exit:

```
li $v0, 10      # exit
syscall
```

1. Two screenshots of the MIPS register panel:  
Before program runs:

The screenshot shows the MIPS register panel in a debugger. The 'Registers' tab is selected, and the 'Coproc 0' sub-tab is active. The register list is displayed with columns for Name, Number, and Value. All registers are initialized to 0, except for the program counter (pc) which is 4194304. The status bar at the top indicates 'Run speed at max (no interaction)'.

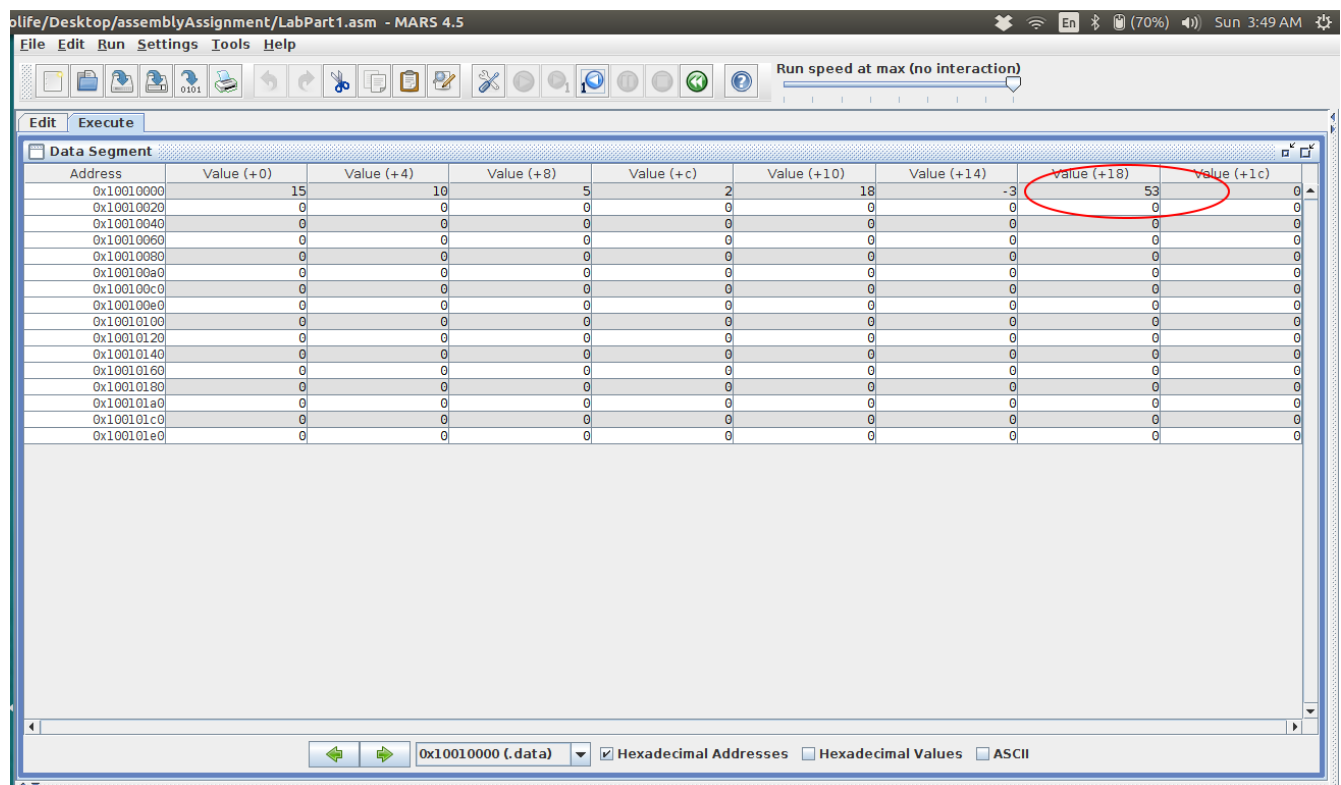
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

After program runs:

The screenshot shows the MIPS register panel after the program has executed. The register values have changed. The program counter (pc) is now 4194436. The status bar at the top indicates 'Run speed at max (no interaction)'.

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	53
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	53
\$t1	9	16
\$t2	10	3
\$t3	11	5
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194436
hi		0
lo		3

Before program runs:



#LabPart2.asm

```
#####  
#                                     #  
#          DTAT SEGMENT                #  
#                                     #  
#####
```

.data

```
A: .word 10    # int A = 10  
B: .word 15    # int B = 15  
C: .word 6     # int C = 6  
Z: .word 0     # int Z = 0
```

```
#####  
#                                     #  
#          TEXT SEGMENT                #  
#          MAIN CODE START NOW        #  
#                                     #  
#####
```

.text

main:

```
lw $t0, A      # load word $t0 = 15  
lw $t1, B      # load word $t1 = 10  
lw $t2, C      # load word $t2 = 6
```

```
#####  
#                                     #  
#          if(A > B || ((C+1) == 7))    #  
#                                     #  
#####
```

```
bgt $t0, $t1, ifAction    # if (A>B) then jump ifAction  
addiu $t3, $t2, 1         # $t3 = (C+1) = $t2 +1  
li $t4, 7                 # $t4 = 7  
bne $t3, $t4, elifCondition    # if ( (C+1)!= 7) = ($t3 != $t4) then jump to elifCondition
```

ifAction:

```
li $t5, 1              # $t5 = 1  
sw $t5, Z              # store word Z = $t5 = 1
```

```
j switch              # unConditional jump to switch
```

```
#####
#                                     #
#     else if(A < B && C > 5)         #
#                                     #
#####
```

elseifCondition:

```
    bgt $t0, $t1, else           # if (A>B) = ($t0 > $t1) jump to else
    li $t4, 5                    # $t4 = 5
    blt $t2, $t4, else          # if (C < 5) = ($t2 < $t4) jump to else

    li $t5, 2                    # $t5 = 2
    sw $t5, Z                    # store word Z = $t5 = 2

    j switch                     # unConditional jump to switch
```

```
#####
#                                     #
#     else                         #
#                                     #
#####
```

else:

```
    li $t5, 3                    # $t5 = 3
    sw $t5, Z                    # store word Z = $t5 = 3
```

```
#####
#                                     #
#     switch(Z)                   #
#                                     #
#####
```

switch:

```
    lw $t4, Z                    # load word $t4 = Z
```

```
##### case 1: #####
```

case1:

```
    li $t5, 1                    # $t5 = 1
    bne $t4, $t5, case2          # if ($t4 != $t5) then jump to case2
    li $t5, -1                   # $t5 = 0 + (-1)
    sw $t5, Z                    # store word Z = $t5
```

```
    j print                      # unConditional jump to print
```

```
##### case 2: #####
```

case2:

```
li $t5, 2          # $t5 = 2
bne $t4, $t5, case3  # if ($t4 != $t5) then jump to case3
li $t5, -2         # $t5 = -2
sw $t5, Z          # store word Z = $t5

j print            # unConditional jump to print
```

```
##### case 3: #####
```

case3:

```
li $t5, 3          # $t5 = 3
bne $t4, $t5, default  # if ($t4 != $t5) then jump to default
li $t5, -3         # $t5 = -3
sw $t5, Z          # store word Z = $t5

j print            # unConditional jump to print
```

```
##### default: #####
```

default:

```
li $t5, 0          # $t5 = 0
sw $t5, Z          # store word Z = $t5
```

```
##### print result #####
```

print:

```
lw $a0, Z          # load word from Z to $a0
li $v0, 1          # print integer value
syscall
```

```
#####
#                  #
# EXIT FROM PROGRAM #
#                  #
#####
```

exit:

```
li $v0, 10         # exit
syscall
```

3. Two screenshots of the MIPS register panel:  
Before program runs:

The screenshot shows the MIPS register panel in a debugger. The 'Registers' tab is selected, and the 'Coproc 0' sub-tab is active. The register list includes \$zero, \$at, \$v0, \$v1, \$a0, \$a1, \$a2, \$a3, \$t0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$s0, \$s1, \$s2, \$s3, \$s4, \$s5, \$s6, \$s7, \$t8, \$t9, \$k0, \$k1, \$gp, \$sp, \$fp, \$ra, pc, hi, and lo. The 'Value' column shows the current state of each register. Most registers are at 0, but \$gp is 268468224, \$sp is 2147479548, and pc is 4194304.

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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194304
hi		0
lo		0

After program finishes:

The screenshot shows the MIPS register panel after the program has finished. The register values have changed. \$at is now 268500992, \$v0 is 10, \$a0 is -1, \$t0 is 10, \$t1 is 15, \$t2 is 6, \$t3 is 7, \$t4 is 1, \$t5 is -1, \$t6 is 0, \$t7 is 0, \$s0 is 0, \$s1 is 0, \$s2 is 0, \$s3 is 0, \$s4 is 0, \$s5 is 0, \$s6 is 0, \$s7 is 0, \$t8 is 0, \$t9 is 0, \$k0 is 0, \$k1 is 0, \$gp is 268468224, \$sp is 2147479548, \$fp is 0, \$ra is 0, pc is 4194528, hi is 0, and lo is 0.

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	-1
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	10
\$t1	9	15
\$t2	10	6
\$t3	11	7
\$t4	12	1
\$t5	13	-1
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194528
hi		0
lo		0



#### 4. Two screenshots of the MIPS memory panel (data tab):

Before program runs:

olife/Desktop/assemblyAssignment/LabPart2.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Stop the currently running program

Edit Execute

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	10	15	6	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

After program finishes:

olife/Desktop/assemblyAssignment/LabPart2.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Stop the currently running program

Edit Execute

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	10	15	6	-1	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

#LabPart3.asm

```
#####
#                                     #
#          DTAT SEGMENT              #
#                                     #
#####
.data

        Z: .word 4          # int Z = 4;
        i: .word 0          # int i = 0;

#####
#                                     #
#          TEXT SEGMENT              #
#          MAIN CODE START NOW      #
#                                     #
#####
.text

main:

        lw $t2, Z            # load word $t2 = Z;
#####
#                                     #
#   for(i=0; i<=21; i=i+3) {Z++;}    #
#                                     #
#####

        lw $t0, i            # load word $t0 = i = 0;
        li $t1, 21           # $t1 = 21

forLoop:

        bgt $t0, $t1, exitForLoop    # if (i>21) = ($t0 > $t1) then exitForLoop
        addiu $t2, $t2, 1             # Z++ = $t2 = $t2 + 1
        addiu $t0, $t0, 3             # i++ = $t0 = $t0 + 1

        j forLoop                    # unConditional jump to forLoop

exitForLoop:

#####
#                                     #
#   do { Z++; } while (Z<100);      #
#                                     #
#####
```

```
li $t1, 100          # $t1 = 100
```

doWhileLoop:

```
addiu $t2, $t2, 1      # Z++ $t2 = $t2 + 1
blt $t2, $t1, doWhileLoop  # if (Z<100) = ( $t2 < $t1 ) then jump to doWhileLoop
```

```
#####
#                               #
#   while(i > 0) { Z--;         #
#               i--; }         #
#                               #
#####
```

whileLoop:

```
blez $t0, exitWhileLoop  # if (i<=0) = ($t0 <= 0) then jump to exitWhileLoop
addiu $t0, $t0, -1        # i-- = $t0 = $t0 + (-1)
addiu $t2, $t2, -1        # Z-- = $t2 = $t2 + (-1)

j whileLoop               # unconditional jump whileLoop
```

exitWhileLoop:

```
sw $t2, Z                # store value from $t2 to Z
sw $t0, i                 # store value from $t0 to i
```

```
##### print result #####
```

print:

```
lw $t2, Z                # load word from Z to $t2
move $a0, $t2             # move $a0 = $t2
li $v0, 1                 # print integer
syscall
```

```
#####
#                               #
#   EXIT FROM PROGRAM         #
#                               #
#####
```

exit:

```
li $v0, 10               # exit
syscall
```

5. One screen shots of the MIPS register panel:  
After program finishes:

**File Edit Run Settings Tools Help**

**Registers Coproc 1 Coproc 0**

Name	Number	Value
\$zero	0	0
\$at	1	268500992
\$v0	2	10
\$v1	3	0
\$a0	4	76
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	100
\$t2	10	76
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194420
hi		0
lo		0

Run speed at max (no interaction)

6. One screenshot of the MIPS memory panel (data tab):  
After program finishes:

olife/Desktop/assemblyAssignment/LabPart3.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	76	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

#LabPart4.asm

```
#####  
#                                     #  
#          DTAT SEGMENT              #  
#                                     #  
#####
```

.data

```
A: .word 0:5          # declare an array of size 5  
B: .word 1, 2, 4, 8, 16 # initialize an array B  
newLine: .asciiz "\n"  # newLine string
```

```
#####  
#                                     #  
#          TEXT SEGMENT              #  
#          MAIN CODE START NOW      #  
#                                     #  
#####
```

.text

main:

```
la $s0, A          # load the address of array A  
la $s1, B          # load the address of array B  
  
move $t0, $s0      # $t0 = $s0 move the address of array A  
move $t1, $s1      # $t1 = $s1 move the address of array B
```

```
#####  
#                                     #  
#    for(i=0; i<5; i++) {           #  
#                                     A[i] = B[i] - 1;      #  
#                                     }                       #  
#####
```

```
li $t2, 0          # i = $t2 = 0  
li $s3, 5          # $s3 = 5
```

forLoop:

```
lw $t4, 0($t1)      # load the value of B[i] to $t4 ( $t4 = B[$t1] )  
  
addiu $t5, $t4, -1   # B[i]-1 = ($t5 = $t4-1)
```

```

sw $t5, 0($t0)          # store the value of $t5 to A[i] = B[i]-1; ( A[$t0] = $t5 )

addiu $t2, $t2, 1       # i++; increment i as $t2 = $t2 + 1

bge $t2, $s3, exitLoop  # if (i>5) = ($t2>$s3) then jump to exitLoop

sll $t6, $t2, 2         # $t6 = $t2 << 2 . multiply the value of $t2 with 4

addu $t0, $s0, $t6      # $t0 = $s0 + $t6 increment the position of array A

addu $t1, $s1, $t6      # $t1 = $s1 + $t6 increment the position of array B

j forLoop               # unconditional jump to fooLoop

```

exitLoop:

```

#####
#           i--;           #
#####

        addiu $t2, $t2, -1    # i--; = ($t2 = $t2 -1) decrement the position of the array
        move $t0, $s0        # move the address of array A $t0 = $s0
        move $t1, $s1        # move the address of array B $t1 = $s1

#####
#           #
#   while(i >= 0) {           #
#           A[i]=(A[i]+B[i]) * 2;           #
#           i--;}           #
#####

```

whileLoop:

```

sll $t4, $t2, 2         # $t4 = $t2 << 2 . multiply the value of $t2 with 4
addu $t0, $s0, $t4      # $t0 = $s0 + $t4
addu $t1, $s1, $t4      # $t1 = $s1 + $t4

lw $t3, 0($t0)          # load the value of A[i] to $t3 ($t3 = A[$t0])
lw $t4, 0($t1)          # load the value of B[i] to $t4 ($t4 = B[$t1])

addu $t3, $t3, $t4      # A[i]+B[i] = ($t3 = $t3 + $t4)

sll $t3, $t3, 1         # (A[i]+B[i])*2; = ($t3 = $t3 < 1) multiply the value of $t3 with 2

sw $t3, 0($t0)          # A[i]=(A[i]+B[i]) * 2; store the value of $t3 to A[i] (A[$t0] = $t3)

```

```

addiu $t2, $t2, -1      # i--; decrement the index of the array ($t2 = $t2 -1)
bltz $t2, exitWhileLoop # if (i<0) = ( $t2< 0 ) then go to exitWhileLoop

```

```

j whileLoop      # unconditional jump to whileLoop

```

exitWhileLoop:

```

##### print the Array A #####

```

print:

```

la $t0, A          # load the address of A to $t0
li $t1, 0          # i=0; load $t0 = 0

```

printLoop:

```

lw $t2, 0($t0)      # A[i] load the value of A[$t0] to $t2
move $a0, $t2        # move the value $t2 to $a0
li $v0, 1            # print the value of $a0
syscall

```

```

la $a0, newLine      # load the address of newLine
li $v0, 4             # print the new line
syscall

```

```

addi $t1, $t1, 1      # i++; ($t1 = $t1 + 1) increment the value of $t1
bgt $t1, 4, exitPrintLoop # if (i>4) = ($t1 > 4) then jump to exitPrintLoop
addi $t0, $t0, 4      # $t0 = $t0 + 4 increase the array position

```

```

j printLoop      # unconditional jump to printLoop

```

exitPrintLoop:

```

#####
#                               #
#      EXIT FROM PROGRAM      #
#                               #
#####

```

exit:

```

li $v0, 10          # exit
syscall

```

After program finishes:

File






Edit











Run

Settings

Tools

Help

Run speed at max (no interaction)

Registers

Coproc 1

Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	1
\$v0	2	10
\$v1	3	0
\$a0	4	268501032
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	268501008
\$t1	9	5
\$t2	10	62
\$t3	11	2
\$t4	12	1
\$t5	13	15
\$t6	14	16
\$t7	15	0
\$s0	16	268500992
\$s1	17	268501012
\$s2	18	0
\$s3	19	5
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194508
hi		0
lo		0

After program finishes:

[illegible]



#LabPart5.asm

```
#####
#                                     #
#          DTAT SEGMENT              #
#                                     #
#####
```

.data

```
i: .word 0                      # int i=0;
inputString: .space 256         # string size 256
hexString: .space 20
ansHexString: .space 20
addressOfM: .word 0             # address of 'm' = char *result = NULL;
letter: .byte 'm'               # letter = 'm'
dialog1: .ascii "First match at address : " # message1
dialog2: .ascii "No match found.\n"        # message2
newline: .ascii "\n\nHexadecimal string: " # newLine
```

```
#####
#                                     #
#          TEXT SEGMENT              #
#          MAIN CODE START NOW      #
#                                     #
#####
```

.text

main:

```
##### take input from user #####
```

```
la $a0, inputString           # load the address of inputString
li $a1, 256                   # the size of the input string is 256
li $v0, 8                     # take input from user
syscall
```

```
move $s0,$a0                  # move the address of inputString to $s0
move $t5,$s0                  # move the address of inputString to $t5
lw $t0, i                     # load the value of the word from i to $t0
lb $t1, letter                 # load the value of the byte from letter to $t1
```

while:

```
lb $t2, 0($t5)                # load the value of the byte from $t5 to $t2
beq $t2, $zero, exitWhile     # if ($t2 == 0) then branch to exitWhile
beq $t1, $t2, storeTheAddress # if ($t1 == $t2 ) then branch to storeTheAddress
addiu $t0, $t0, 1             # i++; ($t0 = $t0 + 1)
```

```

        addu $t5, $s0, $t0                # $t5 = $s0 + $t0

j while                                # unconditional jump to while

##### store the address of 'm' #####

storeTheAddress:

        move $t0, $t5                    # move the address store in $t5 to $t0
        sw $t0, addressOfM                # store the address of 'm' in the inputString to
addressOfM
        lw $t0, addressOfM                # load the address of 'm' from addressOfM

        la $a0, dialog1                  # load the address of dialog1
        li $v0, 4                        # print the string
        syscall

        move $a0, $t0                    # move the value of $t0 to $a0 to print
        li $v0, 1                        # print the integer value in $a0
        syscall

##### conver the address of decimal to hexadecimal #####

        li $t1, 16                      # load imidiata value $t1 = 16
        li $t7, 0                      # load immediate $t7 = 0

loop:

        div $t0,$t1                      # divide $t0 / $t1
        mfhi $t2                        # move the reminder to $t2
        mflo $t3                        # move the quoitent to $t3

        beqz $t0, loopExit                # if ($t0 == 0 ) then branch to loopExit
        move $t0, $t3                    # move value of $t3 to $t0
        bge $t2, 10, setChar              # if ($t2>=10) then branch to setChar
        addiu $t3, $t2, 48                # $t3 = $t2 + 48

        j initilize                      # unconditional jump to initilize the hexString

setChar:

        addiu $t3, $t2, 55                # $t3 = $t2 + 55

initilize:

        sb $t3, hexString($t7)           # store $t3 byte to hexString at ($t7) position
        addiu $t7, $t7, 1                 # $t7 = $t7 + 1 increase the index

```

```
j loop                # unconditional jump to loop
```

```
loopExit:
```

```
    li $t2, 0          # load immediate $t2 = 0
    sb $t2, hexString($t7)  # store $t2 byte to hexString at ($t7) position

    addiu $t7, $t7, -1  # $t7 = $t7 -1 decrement the index
    addiu $t0, $zero, 0 # set the value of $t0 = 0
```

```
reverseTheHexString:
```

```
    blt $t7, $zero, print  # if ($t7 < 0) then jump to print label
    lb $t2, hexString($t7)  # load byte from hexString to $t2
    sb $t2, ansHexString($t0) # store byte from $t2 to hexString
    addiu $t0, $t0, 1        # $t0 = $t0+1 increment the index of ansHexString
    addiu $t7, $t7, -1       # $t7 = $t7 -1 decrement the index of hexString
```

```
j reverseTheHexString    # unconditional jump to reverseTheHexString
```

```
print:
```

```
    li $t2, 0          # set the value of $t2 = 0
    sb $t2, ansHexString($t0) # store $t2 byte to ansHexString of $t0 position

    la $a0, newline     # load the address of newLine
    li $v0, 4            # print the newLine string
    syscall

    la $a0, ansHexString # load the address of ansHexString
    li $v0, 4            # print the ansHexString string
    syscall
```

```
j exit
```

```
exitWhile:
```

```
    la $a0, dialog2     # load the address of dialog2
    li $v0, 4            # print the string
    syscall
```

```
#####
#                                     #
#      EXIT FROM PROGRAM              #
```

```
#                                     #
#####
```

exit:

```
li $v0, 10          # exit from program
syscall
```

9. One screenshot of the MIPS memory panel (data tab) after your program finishes:

The screenshot shows the MARS 4.5 MIPS simulator interface. The main window displays the Data Segment memory panel, which is a table with columns for Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c). The memory addresses range from 0x10010000 to 0x100101e0. The values are mostly 0x00000000, but there are some non-zero values at specific addresses, such as 0x6d647361 at 0x10010004 and 0x10010007. The Registers panel on the right shows the state of the MIPS registers, including \$zero, \$at, \$v0, \$v1, \$a0, \$a1, \$a2, \$a3, \$t0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$s0, \$s1, \$s2, \$s3, \$s4, \$s5, \$s6, \$s7, \$t8, \$t9, \$k0, \$k1, \$gp, \$sp, \$fp, \$ra, pc, hi, and lo. The \$v0 register is highlighted with a red circle, showing a value of 10. The Registers panel also shows the Coprocessor 1 and Coprocessor 0 registers.

Below the Data Segment panel, the Mars Messages panel shows the output of the program. It displays the message "First match at address : 268500999" and "Hexadecimal string: 10010007". The message "program is finished running" is also visible.

Life/Desktop/assemblyAssignment/LabPart5.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0 \0 \0 \0	r k g s	e o m g	\n o k g	\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
0x100100e0	\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
0x10010100	\0 \0 \0 \0	0 0 0 9	1 0 0 1	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	1 0 0 1	9 0 0 0
0x10010120	\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
0x10010140	r d d a	s s e	N \0	a m o	h c t	n u o f	\0 \n . d	e H \n \n
0x10010160	e d a x	a m i c	t s l	g n i r	\0 \0	:	\0 \0 \0 \0	\0 \0 \0 \0
0x10010180	\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
0x100101a0	\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
0x100101c0	\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
0x100101e0	\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

Mars Messages Run I/O

sgkrmoegko  
First match at address : 268501001

Clear Hexadecimal string: 10010009  
-- program is finished running --

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x0000000a
\$v1	3	0x00000000
\$a0	4	0x10010118
\$a1	5	0x00000100
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000008
\$t1	9	0x00000010
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x10010009
\$t6	14	0x00000000
\$t7	15	0xffffffff
\$s0	16	0x10010004
\$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	0x7ffefffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400148
hi		0x00000000
lo		0x00000000

- \* Best part is we know about assembly language
- \* I don't think so;
- \* In future you must provide more exam and assignment.

