Homework 8: Assembly Language

CS 200 • 10 Points Total
Due Wednesday, April 5, 20167

Assignment

Work the following problems.

1. QTSPIM is loaded with the following program in the user text section. It will, after running the built-in kernel, start at the instruction at address 0040024. The beginning user data segment is shown after the code. Pretend you are QTSPIM and execute the program until you reach the end. Keep track of any changes to the registers and user data segment. After you are done, show the contents of the user data segment and registers $t0 - $t7. A blank form is given on the next page. (6 pts.)

   If you are not used to QTSPIM, know that it comments your original code with a ';' but it also adds its own comments inside '[ ]'. So in the 7th line, [Addr1] is just a comment that can be ignored.

```
[00400024] 3c010040  lui $1, 64               ; 19: li $t5, 0x400024
[00400028] 342d0024  ori $13, $1, 36
[0040002c] 340e006c  ori $14, $0, 108         ; 20: li $t6, 0x6C
[00400030] 01cd7021  addu $14, $14, $13       ; 21: addu $t6, $t6, $t5
[00400034] 340f0040  ori $15, $0, 64          ; 22: la $t7, 0x40
[00400038] 01ed7821  addu $15, $15, $13       ; 23: addu $t7, $t7, $t5
[0040003c] 3c0b1001  lui $11, 4097 [Addr1]    ; 24: la $t3, Addr1
[00400040] 256b0004  addiu $11, $11, 4        ; 25: addiu $t3, $t3, 4
[00400044] 3c011001  lui $1, 4097             ; 26: lw $t0, Num1
[00400048] 8c280004  lw $8, 4($1)
[0040004c] 3c011001  lui $1, 4097             ; 27: lw $t1, Num2
[00400050] 8c290008  lw $9, 8($1)
[00400054] 01096020  add $12, $8, $9          ; 28: add $t4, $t0, $t1
[00400058] ad6c0018  sw $12, 24($11)          ; 29: sw $t4, 24($t3)
[0040005c] 01e00008  jr $15                   ; 30: jr $t7
[00400060] 01c00008  jr $14                   ; 31: jr $t6
[00400064] 0109001a  div $8, $9               ; 32: div $t0, $t1
[00400068] 00006010  mfhi $12                 ; 33: mfhi $t4
[0040006c] 3c011001  lui $1, 4097             ; 34: sw $t4, DivHi
[00400070] ac2c000c  sw $12, 12($1)
[00400074] 00006012  mflo $12                 ; 35: mflo $t4
[00400078] 3c011001  lui $1, 4097             ; 36: sw $t4, DivLo
[0040007c] ac2c0010  sw $12, 16($1)
[00400080] ad680014  sw $8, 20($11)           ; 37: sw $t0, 20($t3)
[00400084] 0128082a  slt $1, $9, $8           ; 38: bgt $t0, $t1, l1
[00400088] 1420fff6  bne $1, $0, -40 [l1-0x00400088]
[0040008c] ad690014  sw $9, 20($11)           ; 39: sw $t1, 20($t3)
[00400090] 03e00008  jr $31                   ; 40: jr $ra
```

User data segment [10000000]..[10040000]

```
[10000000]..[10010003]  00000000
[10010004]  00005000  00003000  00000000
[10010020]..[1003ffff]  00000000
```

$t0	5000
$t1	3000
$t2	0
$t3	10010004
$t4	1
$t5	400024
$t6	400090
$t7	400064

User data segment [10000000]..[10040000]

[10000000]..[10010003]	00000000			
[10010004]	00005000	00003000	00002000	
[10010010]	00000001	00000000	00005000	00008000
[10010020]..[1003ffff]	00000000			

2. Write a MIPS assembly implementation of the following C/C++ code. Assume small, unsigned integer arithmetic (no range checking or dealing with negatives needed). Also assume variable names are defined in .data, so you will need to load/store them appropriately. (2 pts. each)

example:

```
if (Num1 > Num2)
   Result = Num1;
else
   Result = Num2 + 5;
```

solution:

```
        lw      $t0, Num1
        lw      $t1, Num2
        bgt     $t0, $t1, N1G
        addiu   $t0, $t1, 5        # Num2 greater, so replace Num1
N1G:    sw      $t0, Result
```

a.

```
if (Num1 < Num2)
     Result = Num1;
else if (Num1 = Num2)
     {
          if (Num2 <= Num3)
             Result = Num2 & Num3;
           else
             Result = Num1 | Num3;
     }
else Result = Num3 + 5;
```

```
        lw      $t0, Num1
        lw      $t1, Num2
        lw      $t2, Num3
        move    $t4, $t0           # assume result is Num 1
        blt     $t0, $t1, end      # if Num1 < Num2, save result
        beq     $t0, $t1, eq12     # go to nested 'if' if Num1 = Num2
        addi    $t4, $t2, 5        # otherwise result is Num3 + 5
        j       end

eq12:   ble     $t1, $t2, le23     # set result if Num2 <= Num3
        or      $t4, $t0, $t2      # else result is Num1 | Num3
        j       end

le23:   and     $t4, $t1, $t2      # result is Num2 & Num3

end:    sw      $t4, Result        # save result
```

b.

```
While( (Num1 < Num2) && (Num2 < Num3))
{
    Num1++;
    Num3--;
}
Result = Num1 + Num2 * Num3;
```

```
        lw      $t0, Num1
        lw      $t1, Num2
        lw      $t2, Num3
loop:
        bge     $t0, $t1, end       # if Num1 >= Num2, end loop
        bge     $t2, $t3, end       # if Num2 >= Num3, end loop
        addi    $t0, $t0, 1         # increment Num1
        addi    $t2, $t2, -1        # decrement Num3
        j       loop

end:    mult    $t1, $t2            # first multiply Num2 * Num3
        mflo    $t4                 # assume it will fit in LO
                # you could also do: mul $t4, $t1, $t2 instead
        add     $t4, $t4, $t0       # add Num1
        sw      $t4, Result         # save result
```