CENG 3420 Homework 1 Solutions

A1
1. a. 0.97; b. 0.92.
2. a. 0.12; b. 0.16.
3. a. 0.97; b. 0.93.

A2
1. The IPS for each processor: 2e9/1.2 = 1.67e9 , 3e9/0.8=3.75e9, 4e9/2.0 = 2.00e9.
2. P1: 2.0e10 cycles, 1.67e10 ins; P2: 3.0e10 cycles, 3.75e10 ins, 4.0e10 cycles, 2.00e10 ins.
3. 1/0.7 = (1+x)/1.2, x = 0.71, so clock rate need to be increased by 71%.

A3
1. $P = \alpha f v^2 C$, $0.76 = 4 * 0.8^2 C_2 / 3 * 1.1^2 C_1$, $\rightarrow C_2/C_1 = 1.0777$.
2. $4 * 0.8^2 C / 3 * 1.1^2 C = 0.70523$
3. $4 * 0.8 * v_2^2 / 3 * v_1^2 = 0.8$, so we get $v_2 = 0.866 v_1$

A4
1.
```
lw $s0, 24($s6)
 sub $s0, $zero, $s0
 sub $s0, $s0, $s1
```
2. 4 registers.

A5
1. f = 16*j + i + g
2. no change
3. 5
```
sll $s0, $s4, 4
add $s0, $s0, $s3
add $s0, $s0, $s1
```

4 needed

A6 Reference answer

1.
```
MAIN: addi $sp, $sp, -4
      sw   $ra, ($sp)
      addi $a0, $0, 10
      addi $a1, $0, 20
      addi $a2, $0, 30
      jal  FUN
      add  $t2, $v0, $0
      lw   $ra, ($sp)
      addi $sp, $sp, 4
      jr   $ra
 FUN: addi $sp, $sp, -8
      sw   $ra, 4($sp)
      sw   $s0, 0($sp)  # save $s0
      lw   $s0, ($s0)   # assume $s0 global varaible
      add  $s0, $s0, $a0
      add  $s0, $s0, $a2
      sub  $v0, $s0, $a1
      lw   $s0, ($sp)
      lw   $ra, 4($sp)
      addi $sp, $sp, 8
      jr   $ra
```

2. After entering MAIN: the stack has $ra in top. After entering FUN, the stack has $ra and $s0 according to this code. It's OK to have other solution.

3. If we can use $t registers in *my_function*, and there's need to be saved them to stack. So we only need to decrease $sp by 4 in *my_function*, not 8 like that in 1.

```
FUN: addi $sp, $sp, -4
     sw $ra, ($sp)
     lw $t0, ($s0)    # assume $s0 is global varaible
     add $t0, $t0, $a0
     add $t0, $t0, $a2
     sub $v0, $t0, $a1

     lw $ra, 0($sp)
     addi $sp, $sp, 4
     jr $ra
```

A7 1. 0x00000012

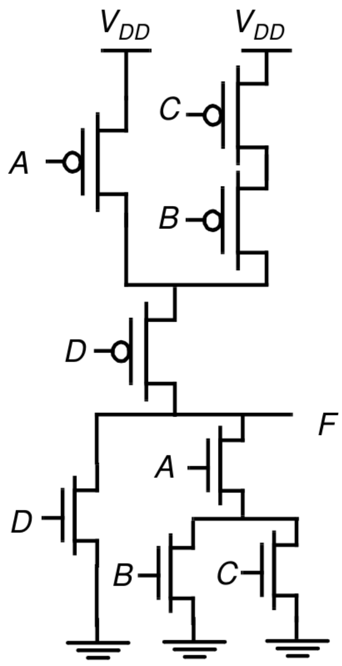
2. 0x00000080

A8 Shown as follows:

a0 = address of the 1st char of the input string
t0 = ASCII value of char “k”, which is 107
t1 = reg. stored the data fetched by the “lw” instr.
t2 and t3 = set regs. for two set statements
v0 = reg. storing the finial address of the data

```
findk:   addi    $t0, $zero, 107        #set the comparison target(“k”)
loop:    lw      $t1, 0($a0)            #load the data form memory to reg.
         seq     $t2, $t1, $zero        #if it is EOL
         bne     $t2, $zero, finish     #jump to finish
         seq     $t3, $t1, $t0          #if it is the target
         bne     $t3, $zero, finish     #jump to finish also
         addi    $a0, $a0, 4            #not the above, move to the next char
         j       loop                   #repeat the loop
finish:  move    $v0, $a0               #move the address in $a0 to $v0
         jr      $ra                    #back to caller of findk
```

A9 As shown below (parallel NMOS for OR and parallel PMOS for AND):



A10
1. a. jump register; b. beq.
2. a. R-type; b. I-type.
3. a. Can jump to any 32b address but need to load a register with a 32b address, which could take multiple cycles.
   b. Allows the PC to be set to the current PC + 4 +/– BranchAddr, supporting quick forward and backward branches but range of branches is smaller than large programs