TA: Jade Cheng

ICS 312

Homework Solution #6

Due Date: September 10, 2009

Exercise #5.1

Question: Translate the following high-level code fragments into assembly language program gragments.

a. `if (A <= B) then Max = B else Max = A`

Answer:

```
                mov     ax, A
                cmp     ax, B
                jle     MaxisB
                mov     Max, ax
                jmp     Done
  MaxisB:
                mov     ax, B
                mov     Max, ax
  Done:
```

b.

```
if (Minute == 59) then
      Minute = 0
      Hour = Hour + 1
else
      Minute = Minute + 1
```

Answer:

```
                cmp     Minute, 59
                je      Minuteis59
                inc     Minute
                jmp     Done
  Minuteis59:
                mov     Minute, 0
                inc     Hour
  Done:
```

c. `if (C == ‘ ’) or (C == 9) then WhiteSpace = 1`

Answer:

```
                cmp     C, ‘ ’
```

```
                    je      WhitSpace
                    cmp     C, 9
                    je      WhitSpace
                    jmp     Done
  WhiteSpace:
                    mov     WhiteSpace, 1
  Done:
```

d.
```
if (C >= ‘A’) and (C <= ‘Z’) then
      UpperCase = 1
else
      UpperCase = 0
```

Answer:
```
                    cmp     C, ‘A’
                    jl      NotUpperCase
                    cmp     C, ‘Z’
                    jg      NotUpperCase
                    mov     UpperCase, 1
                    jmp     Done
  NotUpperCase:
                    mov     UpperCase, 0
  Done:
```

e.
```
OrdTime = MilTime;
if (MilTime < 1200) then {
      AMPM = ‘A’
      if (MilTime < 100) then
            OrdTime = OrdTime + 1200
}
else {
      AMPM = ‘P’
      if (MilTime >= 1300) then
            OrdTime = OrdTime - 1200
}
```

Answer:
```
                    mov     ax, MilTime
                    mov     OrdTime, ax
                    cmp     MilTime, 1200
                    jge     Else
                    mov     AMPM, ‘A’
                    cmp     MilTime, 100
                    jge     Done
                    add     OrdTime, 1200
```

```
                jmp     Done
        Else:
                mov     AMPM, 'P'
                cmp     MilTime, 1300
                jl      Done
                sub     OrdTime, 1200
        Done:
```

Question: Translate the following decorated pseudo-code statement into assembly language:

```
if (A >= 14) or (B == 127) and (C != -5) then X = X + 1
```

Answer:

```
                cmp     A, 14
                jge     True
                cmp     B, 17
                jne     False
        True:
                cmp     C, -5
                je      False
                inc     X
        False:
```

Question: Each of the following assembly language fragments corresponds to a single high-level language `if` statement. Give such a statement in pseudo-code or the high-level language of your choice (Note that you shouldn't use register names in the high-level language version, where they would be meaningless.)

a.

```
                cmp     A, 1
                jnl     A1
                mov     A, 1
        A1:
```

Answer:

```
if (A < 1) then A = 1
```

b.

```
                mov     ax, A
                cmp     ax, B
                jng     B1
                sub     B, ax
                jmp     B2
        B1:
                mov     ax, B
                sub     A, ax
        B2:
```

Answer:

```
if (A <= B) then
      A = A – B
else
      B = B -A
```

c.

```
                cmp     A, 1
                je      D1
                cmp     A, 2
                jne     D2
  D1:
                inc     A
                jmp     D3
  D2:
                dec     A
  D3:
```

Answer:

```
if (A == 1) then
      A++
else if(A != 2)
      A--
```

Question: Eliminate jumps around jumps in the following code fragments:.

a.

```
                cmp     A, 14
                jge     S1
                jmp     S2
  S1:
                mov     X, 21
  S2:
```

Answer:

```
                cmp     A, 14
                jl      Done
                mov     X, 21
  Done:
```

b.

```
                cmp     X, ax
                jne     L1
                jmp     L2
  L1:
                neg     X
                inc     X
  L2:
```

Answer:

```
                    cmp     X, ax
                    je      Done
                    neg     X
                    inc     X
          Done:
```

Question: The actual test performed by the `jg` instruction is 'jump if `ZF = 0` and `SF = OF`.' It should be clear that the condition `ZF = 0` is necessary—it means that `op1 ≠ op2`. Also, `SF = OF = 0` means that there was no overflow on the subtraction and the sign what positive, so `op1≥ op2`. But what is the meaning of `SF = OF = 1`, and why is it necessary to include it as one of the conditions under which `op1>op2`?

Answer: `ZF = 0`, indicates `op1 ≠ op2`. `SF = OF = 1`, indicates the sign bit and the overflow bit are both set. The sign bit is set shows it appears to be a negative number. The overflow bit is set shows whatever symbol (positive or negative) it appears is not the actual symbol. Therefore, we learn from the combination: `SF = OF = 1`, that the subtraction of the comparison yields a positive number. Therefore the left side of the comparison is greater than the right side of the comparison. This is the same conclusion of `ZF = 0`, and `SF = OF = 0`.

Overall, for signed number comparisons, left side is greater than right side, when `ZF = 0`, and `SF = OF`; Left side is smaller than the right side when `ZF = 0`, and `SF ≠ OF`; Left side equals to the right side, when `ZF = 0`.

Question: What is the actual test performed on the flags by the `jl` instruction?

Answer: The actual test performed on the flags by the `jl` instruction is to look at `ZF`, `SF`, and `OF`. It requires `ZF = 0`, and `SF ≠ OF`. If the flags are in this condition, `jl` would lead to a jump.

Exercise #5.3

Question: Rewrite the following assembly code segments without using the `cmp` instruction.

a.

```
sub     al, 'a'
cmp     al, 0
jl      notLc
```

Answer:

```
sub     al, 'a'
jl      notLc
```

b.

```
add     bx, 100
cmp     bx, 0
jne     NotM100
```

Answer:

```
add     bx, 100
jne     NotM100
```