16.317: Microprocessor Systems Design I

Fall 2015

Homework 3 Solution

Each of the questions on the next two pages asks you to convert a high-level code sequence to x86 assembly. Please note that each of these code sequences is part of a larger function—do not worry about any of the function call and stack frame details discussed during lectures 14-16. (That material will be covered on HW 4.)

1. *(25 points) Implement the following conditional statement. You may assume that "X" and "Y" refer to 16-bit variables stored in memory, which can be directly accessed using those names (for example, `MOV AX, X` would move the contents of variable "X" to `AX`).*

```
if (AX < 10) {
   CX = X + 10;
}
else if (AX == 20) {
   CX = CX - Y;
}
else {
   CX = X + Y;
}
```

Solution: Other solutions may be acceptable; the key pieces to this problem are:

- Evaluating the two conditions properly
- Ensuring that you only execute one of the blocks—the "if" case, "else if" case, or "else" case.

```
     CMP  AX, 10
     JL   IF         ; Go to "IF" if AX < 10
     CMP  AX, 20
     JE   ELIF       ; Go to "ELIF" if AX == 20
     MOV  CX, X      ; "Else" case--CX = X + Y
     ADD  CX, Y
     JMP  FIN        ; Skip "if", "else if" cases
IF:  MOV  CX, X      ; "If" case--CX = X + 10
     ADD  CX, 10
     JMP  FIN        ; Skip "else if" case
ELIF: SUB CX, Y      ; "Else if" case--CX = CX - Y
FIN:                 ; End of statement
```

2. *(25 points) Implement the following loop. As in question 1, assume "X" is a 16-bit variable in memory that can be accessed by name. (Hint: Any loop that executes the correct number of iterations is acceptable—you do not necessarily have to change your loop counter in exactly the same way as the for loop, since i is not used in the body of the loop.)*

```
for (i = 0; i < X; i++) {
   AX = AX + X;
   BX = BX - X;
   if (AX == BX)
        break;          // Exit loop early
}
```

Solution: Other solutions may be valid; the key pieces of this problem are:

- Ensuring that the assignment statements are enclosed in a loop with X iterations.
  - Note that, as mentioned above, any loop with X iterations will be valid. The solution below takes advantage of the x86 LOOP instructions so that the actual loop counts from X down to 0, rather than counting up.
- Comparing AX to BX and exiting the loop early if they are equal.
  - Note that this can be accomplished by using a LOOPNE instruction, as shown below, or by adding an explicit jump instruction that leaves the loop when the condition is true.

```
     MOV  CX, X          ; CX = X = # of loop iterations
L:   ADD  AX, X          ; AX = AX + X
     SUB  BX, X          ; BX = BX - X
     CMP  AX, BX
     LOOPNE L            ; Decrement CX, then check if
                         ;   CX is non-zero and previous compare
                         ;   result is "not equal" (AX != BX)
                         ; If either of those conditions are
                         ;   false, exit loop
```

3. *(25 points) Implement the following conditional statement. As in question 1, assume "X" and "Y" are 16-bit variables in memory that can be accessed by name. (Note: Make sure you carefully count the parentheses to make sure you combine conditions correctly!)*

```
if (((AX < X) && (BX < Y)) || ((AX > Y) && (BX > X))) {
   AX = AX - BX;
}
```

Solution: Other solutions may be possible; the key piece of this problem is the evaluation of the complex condition shown, which can be done with SETcc instructions. Note that a series of jump instructions can also be used to evaluate that condition.

```
     CMP  AX, X
     SETL DL         ; (AX < X)
     CMP  BX, Y
     SETL DH         ; (BX < Y)
     AND  DL, DH     ; ((AX < X) && (BX < Y))
     CMP  AX, Y
     SETG CL         ; (AX > Y)
     CMP  BX, X
     SETG CH         ; (BX > X)
     AND  CL, CH     ; ((AX > Y) && (BX > X))
     OR   DL, CL     ; Logical OR of previous complex conditions
                     ; DL is now 1 if the entire condition in the
                     ;   if statement is true
     JZ   SKIP       ; If result of OR is zero, skip subtraction
     SUB  AX, BX     ; AX = AX - BX
SKIP:                ; End of code
```

4. *(25 points) Implement the following loop. As in previous questions, assume "X", "Y", and "Z" are 16-bit variables in memory that can be accessed by name. Recall that a while loop is a more general type of loop than the for loop seen in question 2—a while loop simply repeats the loop body as long as the condition tested at the beginning of the loop is true.*

```
while (X >= Y) {
   Y = Y + Z - 1;
   X = X - Z + 1;
}
```

Solution: Other solutions may be correct.

```
   MOV  DX, Z          ; Z = DX
L: MOV  AX, X          ; AX = X
   CMP  AX, Y          ; Compare X & Y
   JL   FIN            ; Jump to end if X < Y
   ADD  Y, DX          ; Y = Y + DX = Y + Z
   DEC  Y              ; Y = Y - 1 = Y + Z - 1
   SUB  X, DX          ; X = X - DX = X - Z
   INC  X              ; X = X + 1 = X - Z + 1
   JMP  L              ; Return to start of loop
FIN:   ...             ; End of statement
```