Comp 255Q - 1M: Computer Organization Name: _____
Lab #3 - Machine Language Programs for the PDP-8
January 26, 2010

Introduction: In this lab you will write, test, and execute a number of simple PDP-8 machine code programs. Because of its complexity and with the advent of assemblers (programs which translated assembly language program in to machine code) that simplified programming computers, machine code programming is not done anymore although there might be some pedagogical merit in having done it at least once.

Instructions:

The programs to be written will be very simple and will only require a handful of machine code instructions. The nine that we will use (four MRI and five op-code 7 instructions) are given below.

Four MRI instructions (where xxx is indirect bit + memory page bit + 7 bit offset)

1xxx	Twos Complement Add (TAD)	- 12xx or 13xx
2xxx	Increment and Skip on Zero (ISZ)	- 22xx or 23xx
3xxx	Deposit and Clear Accumulator (DCA)	- 32xx or 33xx
5xxx	unconditional Jump (JMP)	- 5200 or 53xx

Five op-code 7 Instructions

7300	Clear Accumulator and Link (CLA CLL)
7041	Complement Accumulator and Increment (CMA IAC or CIA); same as Negate Accumulator
7500	Skip next instruction on Minus Accumulator (SMA)
7510	Skip next instruction on Positive (or zero) Accumulator (SPA)
7402	Halt

Operand Addresses:

Although the PDP-8 has four addressing modes, we'll restrict ourselves to using current page (direct) addressing for our MRI instructions. In current page addressing the effective address is obtained by concatenating the page number of the address of the MRI instruction (bits 0...4) with the contents of the offset field of the instruction. This means the operand referenced by the MRI instruction is always on the same page as the instruction.

For example

```
0205/3320                                  address/contents (octal)

000 010 000 101 / 011 011 010 000          address/contents (binary)

00001|0 000 101 / 011|0|1|1 010 000        page offset / MRI format
  ↘                    ↙
00001 + 1010000 = 000 011 010 000          page + MRI offset concatenated

0320                                       Effective Address for instruction
```

Determining the offset value for an instruction from a known effective address is easier assuming current page addressing; simply convert the octal value into binary (12 bits) and use the last 7 (bits 5…11) for the offset value.

Placing Code in Memory

PDP-8 programs have (at least) two segments: a data segment (a contiguous block of words where data is stored) and a code segment. It's important to separate the two as you don't want to execute your data and you don't want to operate on your code.

The standard PDP-8 convention seems to be that code starts at location 0200o (beginning at the upper half page 1) and data goes somewhere below. We'll position data starting at location 0300o (beginning at the lower half of page 1). Since our initial programs are small, code and data will not overlap.

Part 1: In this part we will write, test and execute a PDP-8 machine code program that adds two integers; specifically the program will sum 2 plus 3 and will look like the c program below. All addressing will be current page (direct) addressing.

```
int main()
{
    int a = 2, b = 3, c; // declare variables
    c = a + b;
}
```

1. Start by writing out the machine code for this program using the PDP-8 Machine Code Form. The code will be placed on page 1 beginning at address 0200 (all values are in octal); data will be placed on page 1 beginning at address 0300.

2 When you have written out the machine code, start the PDP-8 Emulator and go to the Debugger Screen to enter the machine code directly into the PDP-8 memory. Use the **P** command **>P200** (enter) to place the memory cursor at address 0200 on page 1. You should see addresses 0200 through 0377 in the memory display (upper right).

Enter the code from the PDP-8 Machine Code Form starting at address 0200. As you enter each 4 digit code hit (Enter). This will automatically advance the memory cursor.

Use the arrow keys to move the cursor to address 0300 and enter the data values the same way.

Before continuing check that the program and code/data you entered agrees with your code.

3. Trace (single step) the execution of this code by positioning the *memory cursor* on the location of the first instruction, address 0200 (this also sets the Program Counter to 0200) and use the *Space Bar* to single step through the code. Hit the Space Bar 5 times noting how the PC, AC, and IR registers change.

4. Enter different values for a and b and re-execute the program. Use the values below.

```
0300/0022
0301/0017
0302/0000
```

Position the memory cursor at location 0200 (i.e. set the PC equal to 0200) and hit the G (Go) key. What happened? Try it a third time with

```
0300/7776
0301/0002
0302/0000
```

Explain what happened? (Hint: Check the Link bit) _____

5. **Save a copy of your machine code:** Use the **w** - Write command (see below) to save your code to a file. Call your file `lab03a.obj`. The code is saved in *address/contents* format (i.e. nnnn/mmmm). When saved, open your file using NotePad and print out a copy of the file. You can also attach comments to each line of machine code explaining what each line of machine code does. Be sure to attach your name and date as a comment.

```
>w h:\Comp255\lab03a.obj
```

Note: The **>l** (load) command can read this file back into PDP-8 memory (use `>l h:\comp255\lab03a.obj`). Reset (clear) the memory first (use **>r** for reset) and try it.

Verification #1: _____

Part 2 - Multiplication using a Simple Counting Loop: On page 3.11 of The PDP-8 Emulator User's Manual there is a program that executes a simple counting loop. The program demonstrates how the ISZ and JMP instructions are used to control a loop although the loop itself (addresses 0204 - 0205) is empty.

```
address/contents          description

  0200/7300         Clear Accumulator and Link
  0201/1301         Load Accumulator with N
  0202/7041         Negate
  0203/3300         Deposit to Counter (and Zero Accumulator)
  0204/2300         Increment Counter and Skip on Zero
  0205/5204         Jump to Address 0204 ("Increment ...")
  0206/7402         Halt
  0207/5200         Jump to 0200

  0300/0000         Counter
  0301/0010         N = 10 (octal)
```

The program will loop N times but to do so, the value in N is *first negated* and stored in a second location called Counter. The Increment and Skip on Zero increments the negative value in Count skipping over the Jump instruction when Counter equals 0.

Study this code until you are sure you understand it.

This loop does nothing (the body of the loop is empty). Modify this loop that it does the multiplication P = M * N by added M to P N times (*assuming N is positive*). That is

Clear Accumulator and Link
Deposit to P (and clear accumulator)
Load Accumulator with N
Negate
Deposit to Counter (and clear accumulator)
Add M into Accumulator
Increment Counter and Skip on Zero
Jump to Instruction "Add M"
Deposit to P (and clear accumulator)
Halt
Jump to 0200

Each step above is one machine code instruction.

Note the two instructions that are inserted into the code: **Add M** and **Store P**. The product is accumulated in the Accumulator within the body of the loop and then stored in P after the loop ends. The Jump instruction has also been changed to jump two instructions back, and not one. Note that *efficient* use is made of the “and clear accumulator” feature for the Deposit instruction.

Do the following

1. Using a PDP-8 Machine Code Form write out the complete code first. Code should start at address 0200, data at address 0300. Allocate four words of memory for Counter, N, M, and P using memory locations 0300 – 0302

```
address/contents ; comment description

    0300/0000     ; Counter
    0301/0003     ; N = 3
    0302/0005     ; M = 5
    0303/0000     ; P
```

2. Go back to the PDP-8 simulator. Reset memory and go to page 1.

```
> r
> p 0200
```

3. Entering Your Code

4. **Trace** (single step) the execution of the code (remember your value are all in octal). What is 3*5 base 8?

5. **Save a copy of your machine code file** (`>w lab03b.obj`) then use NotePad to print out a copy of your machine code.

Verification #2: _____

Part 3 - Subtraction: Write a program to subtract b from a putting the result in c (i.e. c = a – b). Since there is no PDP-8 subtract instruction we have to first negate b then add a to it. Again (using the common convention) we will place this code on page beginning at address 0200 for code. Data will be placed beginning at 0300 with a being stored at 0300, b at 0301, and c at 0302.

Do the following

1. Using a PDP-8 Machine Code Form write out the machine code. Note that to compute a - b you must first load b (i.e. add it into a zeroed out accumulator), negate it then add a: thus obtaining $-b+a$. Otherwise your code is similar to Program 1 above.

clear accumulator and link
load b
negate b
add a
store result to c
halt
jump to 0200

As per our convention locate a, b, and c at addresses 0300 – 0302. For data feel free to choose convenient values for a and b but make sure *a is greater than b*. Set c to 0.

2. Use the **r** - Reset command to clear out memory and all registers then use the **p** (PC command) to move the memory cursor to address 0200 and display page 1 of memory

Enter your code starting at address 0200 and the data at starting at address 0300

3. **Trace** (single step) the execution of this code. Position the memory cursor on the location of the first instruction, address 0200 and use the (Space Bar) to single step through the code. Hit the Space Bar 6 times noting how the PC, AC, and IR registers change.

4. **Run Again**: Using different values for a and b execute your program again. Enter the following values into a, b, and c.

```
0300/0017
0301/0023
0302/0000
```

Position the memory cursor at location 0200 (i.e. set the PC equal to 0200) and hit the G (Go) key. What happened? Is this answer correct? Think, what would a negative number look like?

Explain what happened _____ _____

5. **Save a copy of your machine code:**. Use the **w** - Write command (see below) to save your code to a file. Call your file `lab03c.obj`. The code is saved in *address/content* format (i.e. nnnn/mmmm). When saved, open your file using NotePad, attach name and date and print out a copy of the file `h:\Comp255\lab03c.obj`

Verification #3: _____

Part 4: Modify program 3 so that you *always subtract the smaller value from the larger*. This will require using a conditional branch. After subtracting b from a, test to see if the accumulator is positive or zero. If positive or zero *skip over the next instruction;* otherwise execute an instruction to negate the result.

Do the following

1. Using the pseudo-code write out the code on a PDP-8 Machine Code Form. Use the same addresses for a, b, and c.

```
clear accumulator and link
load b
negate b
add a
skip if positive
negate accumulator (making it positive)
store accumulator to c
halt
jump to beginning
```

2. Use the **r** - Reset command to clear out memory and all registers then use the **p** (PC command) to move the memory cursor to address 0200 and display page 1 of memory

Enter your code starting at address 0200 and the data at starting at address 0300

3. Execute your program (use either Go and the trace option) to make sure it works. There are three options here: a < b, a = b, and a > b. Be sure to test all three.

Save a copy of your machine code: Use the **w** - Write command (see below) to save your code to a file. Call your file `lab03d.obj`. The code is saved in *address/content* format (i.e. nnnn/mmmm). When saved, open your file using NotePad, attach name and date, and print out a copy of the file `>w h:\Comp255\lab03d.obj`

Verification #4: _____

Part 5 (with time): There is a small problem with Part 2. N is *assumed to be positive* but what happens if N = 0? (Think!) The "fix" is simple. After you "Load Accumulator with N" and "Negate" it, a strictly positive N should now be negative! Test this! Insert instruction **7500** to skip on minus accumulator! If it is not negative, then N must have been zero (or negative to begin with which *contradicts* the precondition that N is positive) so **Jump** to the **Halt** instruction at the end of the program (leaving P equal to zero); otherwise skip over the jump and continue with the multiplication.

Save a copy of your machine code file (`>w h:\comp255\lab03e.obj`) then use NotePad to print out a copy of your machine code.

Verification #5: _____

Hand In

This lab assignment sheet
Print outs of programs lab03a.obj, lab03b.obj, lab03c.obj, lab03d.obj and lab03e.obj