Homework #2 Solutions

2.2 Refer to the four items on page 35 that an instruction must specify. What would need to be specified by the Intel 8086 instruction JCXZ described in Table 2.3?

**Answer:** The four things that must be specified by an instruction are: (1) which operation to perform, (2) where to find the operands, (3) where to put the result, if there is a result, and (4) where to find the next instructions. The JCXZ instruction is a jump instruction. The J CXZ instruction will first test the content of the CX register. If the register is zero then a jump will occur to an address that is specified as part of the instruction. First, the instruction must specify the operation code (opcode) that tells the control unit that this is a J CXZ instruction. Second, the instruction must specify that the CX register is to be tested for zero. This is often implicitly part of the opcode. Third, the instruction must specify the branch address (the address that will be branched to, if the branch is taken. Finally, the instruction must also specify the next instruction address, if the branch is not taken. This is implicitly defined as the content of the program counter.

2.4 Write the code to implement the expression A = (B – C)\*D on 3-, 2-, 1-, and 0-address machines. Do not rearrange the expression. In accordance with programming language practice, computing the expression should not change the value of its operands.

**Answer:** Assume that you have addition (ADD), subtraction (SUB), multiplication (MPY), and data movement (MOV, LOAD, STORE, PUSH, and POP) instructions available to you in each of the relevant types of machines. Recall that an n-address machine will specify n operand addresses in the instruction. So, a 3-address machine has instructions like ADD X, Y, Z. This instruction will perform M[X] ← M[Y] + M[Z]. In words, this takes the content of the memory location specified by the address Y, adds it to the content of the memory location specified by the address Z, and places the result in the memory location specified by the address X. Also, keep in mind that 1-address machines use an accumulator to hold one source operand and the destination operand, and 0-address machines use a stack to store both source operands and destination operands. Finally, you may assume that a memory location can be both a source and a destination.

Note that in all cases in this solution that we read instructions such that SUB A, B, C will perform A = B – C. Similarly, SUB A, B will perform A = A – B.

The table below shows the four programs that are needed to answer this question.

<table>
<thead>
<tr>
<th>3-Address Machine</th>
<th>2-Address Machine</th>
<th>1-Address Machine</th>
<th>0-Address Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB A, B, C</td>
<td>MOV T1, B</td>
<td>LOAD B</td>
<td>PUSH B</td>
</tr>
<tr>
<td>MPY A, A, D</td>
<td>SUB T1, C</td>
<td>SUB C</td>
<td>PUSH C</td>
</tr>
<tr>
<td></td>
<td>MPY T1, D</td>
<td>MPY D</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>MOV A, T1</td>
<td>STORE A</td>
<td>PUSH D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MPY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>POP A</td>
</tr>
</tbody>
</table>
2.5 Compute the total memory traffic in bytes for both instruction fetch and instruction execution for the code that implements the expression evaluation for the four machines in Exercise 2.4 above. Assume opcodes occupy one byte, addresses occupy two bytes, and data values also occupy two bytes.

Answer: Make sure that you understand that all instructions are in memory and must be fetched from memory. Also, make sure that you understand that all data values are in memory and must be fetched from memory. So, an instruction like SUB A, B, C requires 1 byte for the opcode, 2 bytes for address A, 2 bytes for address B, and 2 bytes for address C for a total of 7 bytes just to represent the instruction. These 7 bytes must be fetched from memory in order to bring the instruction into the processor for decoding and execution. The data at each address (A, B, and C) in memory is 2 bytes each. So, fetching the SUB A, B, C instruction requires reading 7 bytes from memory. During execution you must read 2 bytes for B, read 2 bytes for C, and then write 2 bytes for A. So, you have a total of 13 bytes that must go to or from memory to fetch and execute this one instruction.

The same thought process applied to the other instructions as well. For example, MOV T1, B requires 5 bytes to represent the instruction (1 byte for the opcode, 2 bytes for address T1, and 2 bytes for address B). Fetching and executing requires reading 5 bytes of instruction, reading 2 bytes for B, and then writing 2 bytes for into T1. The total traffic is 9 bytes. The LOAD B instruction requires 1 byte for opcode and 2 bytes for address B. Fetching and executing LOAD B requires 5 bytes of memory traffic. An instruction like SUB only requires 1 bytes of memory traffic because all data operations are performed internal to the computer using the internal stack.

So, the total memory traffic is:

3-Address Machine: 26 bytes
2-Address Machine: 36 bytes
1-Address Machine: 20 bytes
0-Address Machine: 22 bytes

2.9 Repeat exercises 2.4 and 2.5 above for a general register machine and the expression A = (B*C) + (D*E). Assume 8-bit opcodes, 5-bit register numbers, and 24-bit addresses.

Answer: Given the 5-bit register numbers you know that you have 32 registers available, so you can label them as R0 through R31. You should also assume that you have LOAD and STORE instructions that can be used to get information from memory into a register and from a register back into memory. In general register machines (illustrated on page 46 of your book) all arithmetic operations are performed on registers. Only the LOAD and STORE instructions will read or write data to or from memory. So, your program can look as follows. There are many other possible solutions. I have tried for the one that is the easiest to read and understand.
LOAD R1, B ; Load data from memory location B into register R1
LOAD R2, C ; Load data from memory location C into register R2
MPY R3, R1, R2 ; Multiply R1 and R2 and store result in R3
LOAD R4, D ; Load data from memory location D into register R4
LOAD R5, E ; Load data from memory location E into register R5
MPY R6, R4, R5 ; Multiply R4 and R5 and store result in R6
ADD R7, R3, R6 ; Add R3 and R6 and store result in R7
STORE A, R7 ; Store content of R7 into memory location A

Each LOAD instruction requires a minimum of 37 bits to represent it (8 bits for the opcode and 5 bits for one register, and 24 bits for the address). The STORE instruction is the same. The MPY instruction requires 23 bits (8 bits for the opcode and 5 bits for each of three registers). You should round these numbers off to the closest number of bytes because most (if not all) machines will be byte addressable. Reading 1 bit from memory will require reading an entire byte from memory. So, LOAD and STORE instructions will require 5 bytes each. The ADD and MPY instructions will require 3 bytes each.

The memory traffic for a LOAD or STORE instruction is 7 bytes (5 bytes for the instruction and 2 bytes for the data). The memory traffic for the MPY or ADD instruction is 3 bytes (3 bytes for the instruction and 0 data read or written to memory). So, the total memory traffic is 44 bytes for my program (34 bytes to fetch all of the instructions and 10 bytes to read and write all the data).

2.12 In the last two instructions of Table 2.3, which of the five items on page 40 are explicitly specified and which are implicit?

**Answer:** The five items are: (1) operation to be performed, (2) location of first operand, (3) location of second operand, (4) place to store the result, and (5) location of the next instruction to be executed. The last two instructions of Table 2.3 are SOB R4, Loop from the DEC PDP 11 machine and JCXZ Addr from the Intel 8086 machine.

**SOB R4, Loop**
(1) Operation to be performed is explicitly specified in the opcode.
(2) Location of first operand (R4) is explicitly specified.
(3) Second operand is implicitly defined as 1 for the decrement.
(4) Result location is implicitly defined (definition of decrement).
(5) Location of the next instruction is explicit (Loop) if the branch occurs and implicit (current program counter value) if the branch does not occur.

**JCXZ Addr**
(1) Operation to be performed is explicitly specified in the opcode.
(2) Location of first operand (CX) is explicitly specified by definition of the instruction.
(3) There is only one operand.
(4) There is no result to be stored.
(5) Location of the next instruction is explicit (Addr) if the branch occurs and implicit (current program counter value) if the branch does not occur.