CSE 2130

Daniel Delgado Acosta
3-13-22

HW #2

(4.5) b.) (1) Interpret locaction 0 & location 1 as 2's Complement integers.

Address	Data		
0000	0001 1110 0100 0011	$\overset{2's}{\Rightarrow}$	$\boxed{1110\ 0001\ 1011\ 1101}$
0001	1111 0000 0010 0101	$\overset{2's}{\Rightarrow}$	$\boxed{0000\ 1111\ 1101\ 1011}$

15 14 13 12   11 10 9 8   7 6 5 4   3 2 1 0

location 0 integer: $\boxed{7,747}$
location 1 integer: $\boxed{-4,059}$

(2) Interpret location 4 as an ASCII Value.

0100 $\underbrace{0000}_{0}\ \underbrace{0000}_{0}\ \underbrace{0110}_{6}\ \underbrace{0101}_{5} \overset{hex}{\Rightarrow}$ 0065

$\overset{ASCII}{\Rightarrow} \boxed{e}$

(3) Interpret locations 6 & 7 as an IEEE floating point number
location 6 contains #[15:0]. Location 7 contains #[31:16]

0110 1111 1110 1101 0011 $\Rightarrow$ 0000 0110 1101 1001 1111 1110 1101 0011
0111 0000 0110 1101 1001

$\overset{32bit}{\Rightarrow}$ $\overset{+}{0}$ $\overset{13}{0\,0000\,1101}$ 10110011111110110100011 $\overset{IEEE}{\Rightarrow}$ $\boxed{1.10110011111110110100011 \times 2^{-114}}$

(4) Interpret location 0 & location 1 as unsigned integers.

0000 0001 1110 0100 0011 $\overset{\text{unsigned int.}}{\Rightarrow}$ $\boxed{7,747}$
0001 1111 0000 0010 0101 $\Rightarrow$ $\boxed{61,477}$

(4.7) 32-bit Instruction: $\underbrace{\boxed{opcode}}_{6\ bits}\underbrace{\boxed{SR}}_{5bits}\underbrace{\boxed{DR}}_{5bits}\underbrace{\boxed{IMM}}_{16\ bits}$, $\underbrace{60 \text{ opcodes}}_{\overset{bit}{\downarrow}\ 2^6=64}$, $\underbrace{32 \text{ registers}}_{\overset{bit}{\downarrow}\ 2^5=32}$

What is the range of IMM? max: $2^{n-1}-1 = 2^{16-1}-1 = 2^{15}-1 = \boxed{32767}$
min: $-2^{n-1} = -2^{15} = \boxed{-32768}$

(4.9) The FETCH phase of the instruction cycle does two important things. One is that it loads the instruction to be processed next into the IR. What is the other important thing? The other important thing the FETCH phase does is increment PC to point to the next instruction.

Fetch instruction	Decode	Evaluate Address	Execute	Store Result
PC				
IR				
MAR				
MDR				

4.11 Step 1 (Fetch Instruction): The memory address register (MAR) is loaded with the contents of the program counter (PC). Also, the PC register is incremented to point at next instruction.
Step 2 (Decode): The instruction is examined to then figure what the processor should do next.
Step 3 (Evaluate Address): The address of the memory location needed to process the instruction is computed.
Step 4 (Fetch Operand Phase): the source operands are obtained to process the instruction.
Step 5 (Execute): After step 3 & 4 are complete the instruction can be executed.
Step 6 (Store Result): The result of the execution is stored in the designated destination, memory or register.

5.1 Given instructions ADD, JMP, LEA, & NOT, identify whether the instructions are operate instructions, data movement instructions, or control instructions. For each instruction, list the addressing modes that can be used with the instruction.

ADD: Operate Instruction, Used to the immediate addressing mode and register addressing mode
JMP: Control instruction, used in the register addressing mode
LEA: Data Movement Instruction, used in the immediate addressing mode
NOT: Operate instruction, used in the register addressing mode

Memory: 256 locations, each location contains 16-bits.

(5.4) a.) How many bits are required for the address?
Address: $\log_2(256) = \boxed{8 \text{ bits}}$

b.) If we use the PC-relative addressing mode, & want to allow control transfer between instructions 20 locations away, how many bits of a branch instruction are needed to specify the PC-relative offset?
The offset uses 2's comp. representation which means a minimum of 6 bits. Therefore, $\boxed{6\text{-bits}}$ is required.

c.) If a control instruction is in location 3, what is the PC-relative offset of address 10?
New PC-relative offset address = (old PC-relative offset add.) − (PC va
=> relative address $= 10 - (3+1) = \boxed{6}$

(5.8) We want to increase the number of registers that we can specify in the LC-3 ADD instruction to 32. Do you see any problem with that? Explain.
Yes, because 32 registers would then require 5-bits for each. The Opcode requires 3 operands which take up 4 bits. There would not be enough bits for the each scenario.

(5.15) R1: $\underbrace{1110}_{\text{LEA}}\,\underbrace{000}_{\text{DR}}\,\underbrace{0\ 0010\ 0000}_{\text{offset}}$ ← x3121

R2: $\underbrace{0010}_{\text{LD}}\,\underbrace{010}_{\text{DR}}\,\underbrace{0\ 00}_{\text{BR}}\,\underbrace{10\ 0000}_{\text{offset}}$ ← M[x3122], data

R3: $\underbrace{1010}_{\text{LDI}}\,\underbrace{011}_{\text{DR}}\,\underbrace{0\ 001}_{\text{BR}}\,\underbrace{0\ 0000}_{\text{offset}}$ ← M[M[x3123]], data

R4: $\underbrace{0110}_{\text{LDI}}\,\underbrace{100}_{\text{DR}}\,\underbrace{0\ 10}_{\text{BR}}\,\underbrace{00\ 0001}_{\text{offset}}$ ← M[R2+x1], data