Andrew Hedy

CSE 313 MW 12pm

Professor Hou

10/31/2016

Homework 2

Chapter 4:

4.5b.) The following represents a small memory. Refer to the table for the following questions.

|  |  |
| --- | --- |
| Address | Data |
| 0000 | 0001 1110 0100 0011 |
| 0001 | 1111 0000 0010 0101 |
| 0010 | 0110 1111 0000 0001 |
| 0011 | 0000 0000 0000 0000 |
| 0100 | 0000 0000 0110 0101 |
| 0101 | 0000 0000 0000 0110 |
| 0110 | 1111 1110 1101 0011 |
| 0111 | 0000 0110 1101 1001 |

The binary value within each location can be interpreted in many ways. We have seen that binary values can represent unsigned numbers, 2's complement signed numbers, floating point numbers, and so forth.

1. Interpret location 0 and location 1 as 2's complement integers.

The value at location 0 is 0001 1110 0100 0011 with the sign value being positive so it’s already in 2’s complement representation.

1\*20 + 1\*21 + 1\*26 + 1\*29 + 1\*210 + 1\*211 + 1\*212 = 1 + 2 + 64 + 512 + 1024 + 2048 + 4096 = **7747**

The value at location 1 is 1111 0000 0010 0101 and the sign is negative so we must first convert by flipping all bits and adding 1. After flipping the bits at adding 1 we get a new binary value of

0000 1111 1101 1011. Now converting to decimal,

1\*20 + 1\*21 + 1\*23 + 1\*24 + 1\*26 + 1\*27 + 1\*28 + 1\*29 + 1\*210 + 1\*211 = 4059. So the 2’s complement is **-4059**

1. Interpret location 4 as an ASCII value.

The value at location four is 0000 0000 0110 0101

Hexadecimal value for location four is 0065

**The ASCII value for location 4 is e**

1. Interpret locations 6 and 7 as an IEEE floating point number. Location 6 contains number [15:0]. Location 7 contains number [31:16].

The value at location 6 is 1111 1110 1101 0011

The value at location 7 is 0000 0110 1101 1001

Since location 6 contains the last bits and location 7 contains the first bits the 32 bit location is

0000 0110 1101 1001 1111 1110 1101 0011

|  |  |  |
| --- | --- | --- |
| 0 | 00001101 | 10110011111111011010011 |

The first bit is the sign bit which signifies the floating point number is positive.

The next 8 bits are the exponent. 1\*20 + 1\*22 + 1\*23 = 13

The last 23 bits are the fraction. (Decimal) = (sign) (1.fraction) x 2exp – 127

The IEEE floating point representation is **+1.10110011111111011010011 x 2-114**

1. Interpret location 0 and location 1 as unsigned integers.

The value at location 0 is 0001 1110 0100 0011

1\*20 + 1\*21 + 1\*26 + 1\*29 + 1\*210 + 1\*211 + 1\*212 = **7747**

The value at location 1 is 1111 0000 0010 0101

1\*20 + 1\*22 + 1\*25 + 1\*212 + 1\*213 + 1\*214 + 1\*215 = **61477**

4.7.) Suppose a 32-bit instruction takes the following format:

|  |  |  |  |
| --- | --- | --- | --- |
| OPCODE | SR | DR | IMM |

If there are 60 opcodes and 32 registers, what is the range of values that can be represented by the immediate (IMM)? Assume IMM is a 2’s complement value.

26 = 64 so the number of bits the opcodes can represent is 6 bits

25 = 32 so the number of bits the storage register and the destination register can represent is 5 each.

32 bits total with 6 bits for opcodes, 5 bits for the storage register and 5 bits for the destination register. That leaves us with 16 bits for the immediate value.

2’s complement can represent -216-1 to 216-1-1. So anywhere from **-32768 to 32767 which is 65535 values**

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 is it loads the address of the next instruction into the PC. So the PC is incremented by 1.**

4.11.) State the phases of the instruction cycle and briefly describe what operations occur in each phase.

1.) **FETCH: The FETCH phase obtains the next instruction from memory and loads it into the instruction register (IR) of the control unit.**

2.) **DECODE:** **The DECODE phase examines the instruction in order to figure out what the micro-architecture is being asked to do.**

3.) **EVALUATE ADDRESS: This phase computes the address of the memory location that is needed to process the instruction.**

4.) **FETCH OPERANDS: This phase obtains the source operands needed to process the instruction.**

5.) **EXECUTE: This phase carries out the execution of the instruction.**

6.) **STORE RESULT: The final phase of an instruction’s execution. The result is written to its designated destination.**

Chapter 5:

5.4.) Say we have a memory consisting of 256 locations, and each location contains 16 bits.

a.) How many bits are required for the address?

2x = 256 🡪 Log2x = Log256 🡪 x = Log2(256) = **8 bits**

b.) If we use the PC-relative addressing mode, and 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 two opcodes that use PC-relative addressing mode are LD (0010) and ST (0011). The address (bits [8:0]) can only be within +256 to -255 of the LD or ST instruction since the PC incremented before the offset is added.**

c.) If a control instruction is in location 3, what is the PC-relative offset of address 10? Assume that the control transfer instructions work the same way as in the LC-3. **Since the PC increments before the offset the instruction would then be in location 4 before the offset. The PC-relative**

5.5.)

a.) What is an addressing mode? **The addressing modes determine where the operands are located.**

b.) Name three places an instruction’s operands might be located. **Instruction’s operands might be located in a register, in memory, or as an immediate value.**

c.) List the five addressing modes of the LC-3, and for each one state where the operand is located (from part b). **The five addressing modes of LC-3 are immediate, register, direct memory address, indirect memory address, base + offset address. A register operand is located in registers (R0-R7). The operands in direct memory address, indirect memory address, and base + offset are located in memory. As for the immediate operand, that’s located in the instruction.**

d.) What addressing mode is used by the ADD instruction shown in section 5.1.2? **Since ADD only uses registers for its function it would be register addressing mode.**

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. **This will not work. Increasing the number of registers to 32 will need 5 bits. The add instruction requires 3 registers which is 15 bits. Now, adding the opcode which is 4 bits that increases the total to 19 bits. For LC-3 there is only 16-bits allocated for an instruction.**

5.15.) State the contents of R1, R2, R3, and R4 after the program starting at location x3100 halts.

|  |  |
| --- | --- |
| Address | Data |
| 0011 0001 0000 0000 | 1110 001 000100000 |
| 0011 0001 0000 0001 | 0010 010 000100000 |
| 0011 0001 0000 0010 | 1010 011 000100000 |
| 0011 0001 0000 0011 | 0110 100 010 000001 |
| 0011 0001 0000 0100 | 1111 0000 0010 0101 |

|  |  |
| --- | --- |
| 0011 0001 0010 0010 | 0100 0101 0110 0110 |
| 0011 0001 0010 0011 | 0100 0101 0110 0111 |

|  |  |
| --- | --- |
| 0100 0101 0110 0111 | 1010 1011 1100 1101 |
| 0100 0101 0110 1000 | 1111 1110 1101 0011 |

Instruction 1

|  |  |  |
| --- | --- | --- |
| 1110 = LEA | 001 = R1 | 000100000 = PCoffset x20 |

Instruction 2

|  |  |  |
| --- | --- | --- |
| 0010 = LD | 010 = R2 | 000100000 = PCoffset x20 |

Instruction 3

|  |  |  |
| --- | --- | --- |
| 1010 = LDI | 011 = R3 | 000100000 = PCoffset x20 |

Instruction 4

|  |  |  |  |
| --- | --- | --- | --- |
| 0110 = LDR | 100 = R4 | 010 = R2 | 000001 = PCoffset x1 |

Instruction 5

|  |  |  |
| --- | --- | --- |
| 1111 = TRAP | 00000 | 0010 0101 = x25 |

LEA, R1, 0x20

LD, R2, 0x20

LDI, R3, 0x20

LDR, R4, R2, 0x1

.HALT

**R1 = x3121**

**R2 = Contents in R2 loaded into x3122**

**R3 = Contents in R3 loaded into x3123**

**R4 = Contents of R4 loaded into R2 + x1**