LE9.4.1: LD and ST Instructions

0.8571428571428571/1 point (ungraded)

- Summary of Instruction Formats (PDF)
- Beta Documentation (PDF)

When the assembler processes your program, it generates the binary representation for instructions and data, placing the results in consecutive locations in main memory. "." is the name of a special symbol the assembler uses to remember the byte address of the next main memory location to be filled. Initially the value of "." is set to 0, then incremented by 4 each time another 32-bit word is generated and stored in memory.

We can use the assembly assignment operator, "=", to change where the assembler will store generated data, like so:

```
    . = 0x6000
    ADDC(R31,1234,R1)
zz: SUB(R31,R1,R2)
```

In this example , the assembler would place the binary for the `ADDC` instruction at location 0×6000 and the binary for `SUB` at location 0×6004.

The example also demonstrates using the ":" operator to assign a symbolic name to the address of a memory location. The statement "`zz:`" is shorthand for "`zz = .`", which sets the value of the symbol "zz" to the address of the next location to be filled, i.e., the address of the location holding the `SUB` instruction. In this example, the symbol "zz" would have the value 0×6004.

In the following questions, assume that assembler has processed the following statements, which initialize a 4-element array of words starting at memory location 0×2000.

```
       . = 0x2000
array: LONG(0xba5eba11)    // initialize a data word in memory
       LONG(0xc0ffee00)
       LONG(0xdeadbeef)
       LONG(0x12345678)
```

The value of the symbol "array" will be 0×2000. So whenever we write "array" in our programs, it's exactly equivalent to writing "0×2000".

(A) Consider the execution of a single instruction:

```
LD(R31,array,R1)
```

Address of memory location accessed by LD? 0x2000 ✓

Answer: 0x2000

Value left in R1? 0xba5eba11 ✓ Answer: 0xba5eba11

Binary representation of `LD(R31,array,R1)`? 01100000011111100100 ✗

Answer: 0x603f2000

Explanation
The LD instruction computes the memory address by adding the contents of the Ra register (R31, which has the value 0) to the sign-extended literal (value of array = 0x2000): 0 + 0x2000 = 0x2000.
LD will access location 0x2000, which, according the information in the problem statement, has been initialized to 0xba5eba11, so that's the value placed in R1.
The 32-bit binary encoding is

```
      opcode  RC    RA    constant
      011000 00001 11111 0010 0000 0000 0000
```

(B) Consider the execution of a short program, which uses R0 as a *pointer*, i.e., a register that contains a memory address. This program copies the second element of the array into the third element of the array.

```
   ADDC(R31,array,R0)
   LD(R0,4,R1)
   ST(R1,8,R0)
```

Value left in R0? 0x2000 ✓ Answer: 0x2000

Value left in R1? 0xC0FFee00 ✓ Answer: 0xc0ffee00

Address of memory location written by ST? 0x2008 ✓

Answer: 0x2008

Recall that the template for ST is `ST(Rc,const,Ra)`. Notice that the order of the operands has Rc first, followed by const, then Ra. That's because in a store operation, Rc is supplying the source operand (the value to be written into memory) and

Mem[const+Reg[Ra]] is the destination.

Binary encoding of the ST instruction? 0b0110010000100000C ✓

Answer: 0x64200008

Explanation
ADDC adds the Ra register value (R31 = 0) to the constant ("array" = 0x2000) and stores the result (0 + 0x2000 = 0x2000) into the Rc register (R0).
LD will access location 0x2000 + 4 = 0x2004, which, according to the information in the problem statement, has been initialized to 0xc0ffee00, so that's the value placed in R1.
ST will access location 0x2000 + 8 = 0x2008.
The 32-bit binary encoding of ST is

```
      opcode  RC     RA     constant
      011001 00001 00000 0000 0000 0000 1000
```

Submit

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: 9. Designing an Instruction Set / LE9.4

Add a Post

Show all posts by recent activity

LD(Ra,literal,Rc): Reg[Rc] ← Mem[Reg[Ra] + SEXT(literal)] 6
LD(Ra,literal,Rc): Reg[Rc] ← Mem[Reg[Ra] + SEXT(literal)] question B) Value left in R1? Why is the ...

The 32-bit binary encoding is (spoiler) 3
011000 00001 11111 0010 0000 0000 0000 Ra = R31 = 0x00000. So how has it become "11111" her...

Question about ". =" operation 2
When the following program is assembled, I get the expected result: 5 is initially stored in R1, then ...

typo in LE9.4.1 2
"SUBC" in "the binary for SUBC at location 0x6004" is not consistent with "zz: SUB(R31,R1,R2)"

order of arguments 2
I hate that the order of the arguments for the assembler in most cases differs from their order in th...

BASEBALL and COFFE? Hahahaha

array: LONG(0xba5eba11) // initialize a data word in memory LONG(0xc0ffee00) LONG(0xdeadbeef...

2

BIN or HEX?

The last problem states to give **binary** encoding, but the grader expects **hex**

3