LE9.4

Bookmark this page

LE9.4.1: LD and ST Instructions

1.0/1.0 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? 0×2000 ✔

Value left in R1? 0xba5eba11 ✔

Binary representation of `LD(R31,array,R1)`? 0b01100000000111111001 ✔

(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? 0×2000 ✔

Value left in R1? 0xc0ffee00 ✔

Address of memory location written by ST? 0×2008 ✔

Calculator

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? 0b01100100001000000C ✔

Submit

Discussion

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

Hide Discussion

Add a Post

Show all posts by recent activity

☑	Register 31 R31 takes the value 11111 or 00000. Can someone explain which value to use or when to use them?	5
💬	baseball, coffee and deadbeef Enjoying the course a lot so far ... :-)	2

‹ Previous Next ›