LE9.3

Bookmark this page

LE9.3.1: ALUC Instructions

1.0/1.0 point (ungraded)

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

For the Beta instruction sequence shown below, indicate the 32-bit two's complement values of the specified registers after the sequence has been executed by the Beta. The effect of the instructions is cummulative, later instructions use the values stored by earlier instructions.

You can find detailed descriptions of each Beta instruction in the "Beta Documentation" handout -- see link above. Remember that register values and the ALU use a 32-bit two's complement representation.

Hint: You can enter answers in hex by specifying a "`0x`" prefix, *e.g.*, 17 could be entered as "`0x11`". Usually one would enter addresses, values in memory, etc. using hex. You can also use a "`0b`" prefix to enter a binary value, e.g., "0b10001".

```
ADDC(R31,0x1234,R1)
SUBC(R1,-1,R2)
SRAC(R2,8,R3)
SHLC(R2,32,R4)
XORC(R2,0xFFFF,R5)
ANDC(R5,0x0F0F,R6)
```

Value left in R1? 0×1234 ✔

Value left in R2? 0×1235 ✔

Value left in R3? 0×12 ✔

Value left in R4? 0×1235 ✔

Value left in R5? 0xFFFFEDCA ✔

Value left in R6? 0×0d0a ✔

Submit

LE9.3.2: Do we need SUBC?

1.0/1.0 point (ungraded)

A 6.004 student here at MIT has proposed that all SUBC instructions of the form

```
    SUBC(Rx,const,Ry)
```

can be replaced by an ADDC that uses the negative of the constant:

```
    ADDC(Rx,-const,Ry)
```

Is the student on to something? Will this transformation work for all SUBC instructions?

- ○ Yes, the transformation will work for all SUBC instructions
- ◉ No, there are SUBC instructions with no ADDC equivalent

✔

Submit

LE9.3.3: ALUC Instruction Encoding?

1.0/1.0 point (ungraded)

Calculator

Please give the 32-bit binary encoding for each of the ALUC instruction shown below.

Note that when translating the symbolic form of the ALUC instruction into binary, the program (called the "assembler") we run will only use the low-order 16 bits of the value provided by the second operand, even if that value requires more than 16 bits to represent correctly. Our assembler is dumb :(

One other note: we've been using the symbol R0 to represent register 0. The assembler knows to treat the Rx symbols as shorthand for the decimal constant x, e.g., R17 is a more readable alternative to specifying the constant 17.

(A) 32-bit encoding for `ADDC(R31,0x1234,R1)`?

0b1100000000111111000 ✔

(B) 32-bit encoding for `XORC(R1,-17,R0)`?

0b11101000000000001111 ✔

(C) 32-bit encoding for `CMPLEC(R12,40000,R0)`?

0b11011000000011001000 ✔

(D) 32-bit encoding for `ADDC(R1,R2,R3)`?

0b11000000011000010 0 ✔

Submit

Discussion

Hide Discussion

Topic: 9. Designing an Instruction Set / LE9.3

Add a Post

Show all posts by recent activity

- LE9.3.1 — 3
  Please help me to explain the difference between SHRC and SRAC?
- Different between ADDC(Rx,-0×8000,Ry) and SUBC(Rx,0×8000,Ry) ? — 6
  I can't see the difference between both instructions. for SUBC(Rx,0×8000,Ry) it should add 0xFFFF8000, not 0×00008000. thanks i...
- LE9.3.3A — 3
  ADDC(R31, 0×1234, R1) My answer is 0xC0201234 but incorrect. Please help me figure out why i'm wrong.
- LE9.3.3: ALUC Instruction Encoding? part C — 3
  The explanation state that: This is a bit of a trick question since the constant decimal 40000 = 0×9C40 is too large to be correctly re...
- Rule for sign extension — 2
  I probably missed this somewhere in the lecture. Could anyone point me to the place in the lecture where the rule for sign extension, ...

Previous Next

Calculator