MITx 6.004.2x

**Computation Structures 2: Computer Architecture**

Help selfpoised

Course Progress Dates Discussion

Course / 13. Building the Beta / Lecture Videos (39:19)

‹ Previous ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ Next ›

## LE13.2

Bookmark this page

Calculator

## LE13.2.1 New Beta Instructions?

1.0/1.0 point (ungraded)

For each of the following potential additions to the Beta instruction set, fill in the table with the control signal settings needed to execute these instructions on an Beta. Assume the Beta datapath and control signals as shown at the end of the previous video. Please choose "do not care" if the value of control signal doesn't matter when executing the instruction.

(A) **Move if zero**

```
Usage:     MVZ (Ra, Rb, Rc)
Operation: PC ← PC + 4
           if Reg[Ra] == 0 then Reg[Rc] ← Reg[Rb]
```

| | | |
|---|---|---|
| ALUFN | boole unit: select B operand | ✓ |
| ASEL | do not care | ✓ |
| BSEL | 0 | ✓ |
| MOE | do not care | ✓ |
| MWR | 0 | ✓ |
| PCSEL | 0 | ✓ |
| RA2SEL | 0 | ✓ |
| WDSEL | 1 | ✓ |
| WERF | if Reg[Ra]==0 then 1 else 0 | ✓ |

(B) **Move constant if zero**

```
Usage:     MVCZ (Ra, literal, Rc)
Operation: PC ← PC + 4
           if Reg[Ra] == 0 then Reg[Rc] ← SEXT(literal)
```

| | | |
|---|---|---|
| ALUFN | boole unit: select B operand | ✓ |
| ASEL | do not care | ✓ |
| BSEL | 1 | ✓ |
| MOE | do not care | ✓ |
| MWR | 0 | ✓ |
| PCSEL | 0 | ✓ |
| RA2SEL | do not care | ✓ |
| WDSEL | 1 | ✓ |
| WERF | if Reg[Ra]==0 then 1 else 0 | ✓ |

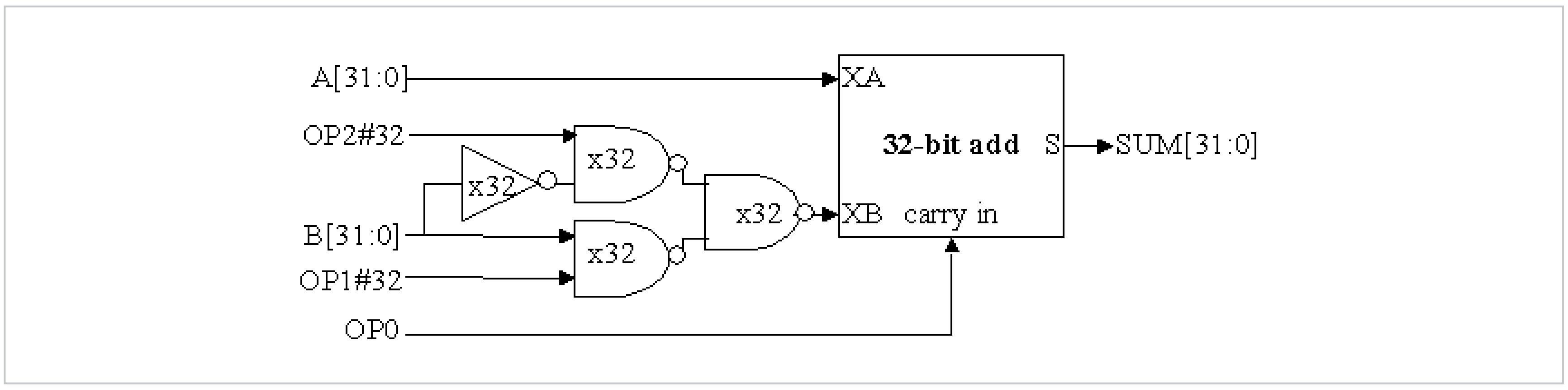
Submit

## LE13.2.2 A new ARITH unit!

Calculator

1.0/1.0 point (ungraded)

Ben Bitdiddle has proposed changing the adder unit of the Beta ALU as shown in the following diagram. His goal is to use the adder unit to compute more than just "A+B" and "A-B". The changes include one additional inverter and three additional 2-input NAND gates for each bit of the adder unit. The "x32" appearing inside the gate icons indicates that those gates are replicated 32 times to handle all 32 bits of incoming data.



(A) For each of the eight possible values of the three control bits OP[2:0] indicate what operation the revised adder unit will perform.

| | | |
|---|---|---|
| OP[2:0] = 0b000 | A | ✓ |
| OP[2:0] = 0b001 | A+1 | ✓ |
| OP[2:0] = 0b010 | A+B | ✓ |
| OP[2:0] = 0b011 | A+B+1 | ✓ |
| OP[2:0] = 0b100 | A-B-1 | ✓ |
| OP[2:0] = 0b101 | A-B | ✓ |
| OP[2:0] = 0b110 | A-1 | ✓ |
| OP[2:0] = 0b111 | A | ✓ |

To show off the capabilities of his new adder unit, Ben proposes adding a LOOP instruction which combines branching and decrementing in a single instruction. Ben's theory is that the SUB/BNE instructions that appear at the end of a FOR-loop can be combined into a single LOOP instruction. Here's his definition for LOOP:

```
Usage:     LOOP(Ra, label, Rc)
Operation: literal = ((OFFSET(label) - OFFSET(current inst))/4) - 1
           PC ← PC + 4
           EA ← PC + 4*SEXT(literal)
           tmp ← Reg[Ra]
           Reg[Rc] ← Reg[Ra] - 1
           if tmp != 0 then PC ← EA
```

The LOOP instruction behaves like a BNE in the sense that it branches if Reg[Ra] is not zero. But instead of saving the PC of the following instruction in Rc, Reg[Ra]-1 is stored in Rc instead. The destination of the branch is determined as for all branches: the literal field of the instruction is treated as a word offset, so it is sign-extended, multiplied by four and added to PC+4 to produce a new value for the PC. Usually Ra and Rc specify the same register.

Consider the following instruction sequence:

```
loop: ADD(R1,R2,R3)
      LOOP(R4,loop,R4)
      ...
```

(B) Suppose R4 is initialized to 8 and then the two-instruction sequence shown above is executed.

How many times will the ADD instruction be executed? 9 ✓

What value is in R4 when "..." is finally executed? Give your answer as a decimal integer.

Calculator

-1 ✓

(C) Fill in the table with the control signal settings needed to execute the LOOP instruction on a Beta that includes Ben's new adder unit. Please choose "do not care" if the value of control signal doesn't matter when executing LOOP.

| | | |
|---|---|---|
| ALUFN | adder unit: A - 1 | ✓ |
| ASEL | 0 | ✓ |
| BSEL | do not care | ✓ |
| MOE | do not care | ✓ |
| MWR | 0 | ✓ |
| PCSEL | if Reg[Ra]==0 then 0 else 1 | ✓ |
| RA2SEL | do not care | ✓ |
| WDSEL | 1 | ✓ |
| WERF | 1 | ✓ |

Submit

## LE13.2.3 Broken WDSEL signals

1.0/1.0 point (ungraded)

The Beta executes the assembly program below starting at location 0 and stopping when it reaches the HALT() instruction. In the following two parts, please give the values in R0 and R1 after the Beta halts. Write the values in hex or write “CAN’T TELL” if the values cannot be determined.

```
    .=0
    LD(R31, i, R0)
    SHLC(R0, 2, R0)
    LD(R0, a-4, R1)
    HALT()
a:  LONG(0xBA5EBA11)
    LONG(0xDEADBEEF)
    LONG(0xC0FFEE)
    LONG(0x8BADF00D)

i:  LONG(3)
```

1. Please indicate the hex values found in R0 and R1 after executing the program above on a fully functioning Beta.

   **Contents of R0 (in hex): 0x** C ✓

   **Contents of R1 (in hex): 0x** C0FFEE ✓

2. Please indicate the hex values found in R0 and R1 after executing the program above on a Beta where the WDSEL[1:0] signal is stuck at the value 1, i.e., 0b01.

   **Contents of R0 (in hex): 0x** 80 ✓

   **Contents of R1 (in hex): 0x** 8C ✓

Calculator

Submit

## Discussion

Hide Discussion

**Topic:** 13. Building the Beta / LE13.2

**Add a Post**

Show all posts ⌄ by recent activity ⌄

☑ Timing strategy to make sure correct order of events?
The data flow analysis in lectures is great, but I think the professor doesn't mention about timing issues, for example those control bi... 3

☑ [STAFF] LE 13.2.2 : A new Arith Unit Part C 8

☑ LE13.2.1 New Beta Instructions
Hello, If we select don't care for RA2SEL to simplify control logic then the assembler must make sure that bit field ID[15:11] and ID[25... 3

☑ LE13.2.3 Part B Solution
Hello! I can't understand why the value stored into R0 when WDSEL is stuck at 1 is 32. If somebody help me, i'll appreciate it very mu... 5

[deleted]
[deleted] 9

‹ Previous Next ›

## edX

About
Affiliates
edX for Business
Open edX
Careers
News

## Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap

## Connect

Blog
Contact Us

Calculator

Calculator