Previous Next

LE13.2

Bookmark this page

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

Explanation
ALUFN: we'll use the ALU to output its B operand. The official answer to use the Boolean operation "select B operand" which is one of the 16 Boolean functions our ALU can compute. But since we're only interested in the ALU result when the A operand (Reg[Ra]) is zero, we could also specify "adder unit: A+B"
ASEL: officially "don't care" when the ALU operation is "select B operand", but should be 0 if the ALUFN is "A+B".
BSEL: 0 to select Reg[Rb] as the B operand.
MOE: don't care since we're not reading from main memory.
MWR: 0 since we're not writing to main memory.
PCSEL: 0 since we're not branching or jumping.
RA2SEL: 0 to select the Rb field of the instruction.
WDSEL: 1 to select the ALU result for write-back.
WERF: uses the Z signal (Reg[Ra]==0) to determine if write-back should occur. We want WERF = Z.

(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	✔ Answer: boole unit: select B operand
ASEL	do not care	✔ Answer: do not care
BSEL	1	✔ Answer: 1
MOE	do not care	✔ Answer: do not care
MWR	0	✔ Answer: 0
PCSEL	0	✔ Answer: 0
RA2SEL	do not care	✔ Answer: do not care

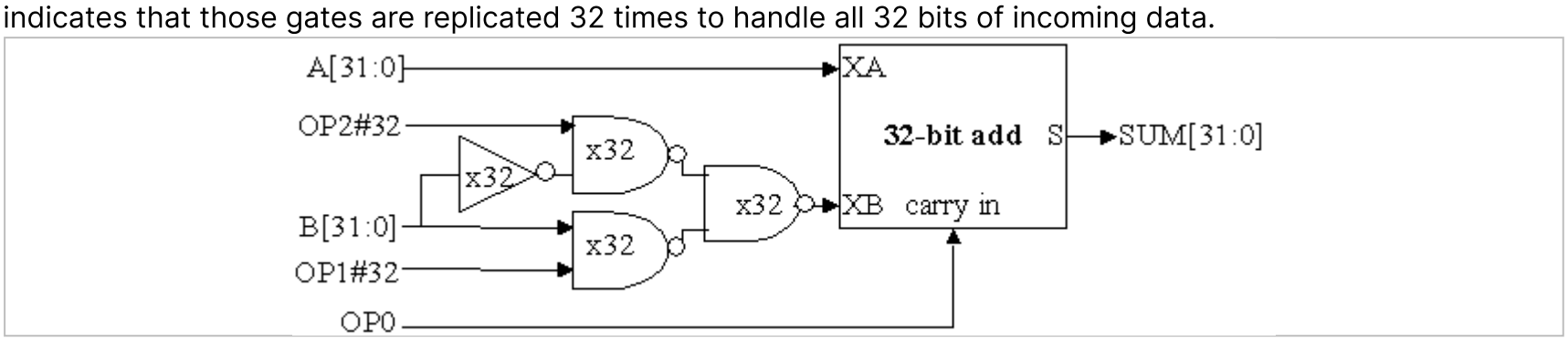
Submit

Answers are displayed within the problem

LE13.2.2 A new ARITH unit!

0.0/1.0 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 Select an option

OP[2:0] = 0b001 Select an option

OP[2:0] = 0b010 Select an option

OP[2:0] = 0b011 Select an option

OP[2:0] = 0b100 Select an option

OP[2:0] = 0b101 Select an option

OP[2:0] = 0b110 Select an option

OP[2:0] = 0b111 Select an option

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)
      ...
```

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

(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	Select an option
ASEL	Select an option
BSEL	Select an option
MOE	Select an option
MWR	Select an option
PCSEL	Select an option
RA2SEL	Select an option
WDSEL	Select an option
WERF	Select an option

Submit

30

LE13.2.3 Broken WDSEL signals

0.0/1.0 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

   Contents of R1 (in hex): 0x

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

Submit

Discussion

Topic: 13. Building the Beta / LE13.2

Hide Discussion

Add a Post

Show all posts by recent activity

- What does "#32" means? — 4
  In the diagram, do "OP1#32" and "OP#32" mean that the operations need to be 32 bits? Why does we need 32 bit op when we only...
- LOOP — 5
  What're the locations of EA and TEMP, and how can we access all of these locations within one CLK cycle is it possible to use the A...
- The value of i is first written into R0, this value is 32 = 0b100000 — 2
  Why is it showing decimal and not hex when it is dealing with memory addresses? Took a while to figure this one out. Same thing in...
- [STAFF]LE13.3.1 Not satisfied with the explanation — 7
  The explanation says RA2SEL is the only one needed before reading the operands from the Regfile. But according to the diagram gi...

Previous Next

Security
Media Kit