

LE9.2.1: ALU Instructions

1/1 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 cumulative, 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".

Hint: It's best to figure out the answers by hand since that will give you practice in understanding Beta assembly language. If you need help, you can copy and paste the code into the BSim Standbox in the Overview section and simulate its execution step-by-step.

```

ADD(r31, r31, r0)
CMPEQ(r0, r31,
r1)
ADD(r1, r1, r2)
OR(r2, r1, r3)
SHL(r2, r3, r4)
SUB(r1, r2, r5)
CMPLT(r31, r5, r6)
SHR(r5, r4, r7)
SRA(r5, r4, r8)

```

Value left in R0?	0x00	✓ Answer: 0
Value left in R1?	0x01	✓ Answer: 1
Value left in R2?	0x02	✓ Answer: 2
Value left in R3?	0x03	✓ Answer: 3
Value left in R4?	0x10	✓ Answer: 16
Value left in R5?	0xFFFFFFFF	✓ Answer: -1
Value left in R6?	0x0	✓ Answer: 0
Value left in R7?	0xFFFF	✓
Answer: 0xFFFF		
Value left in R8?	0xFFFFFFFF	✓
Answer: 0xFFFFFFFF		

Explanation

ADD(r31, r31, r0) adds R31 (with a value of 0) to R31, giving 0, which is then stored in R0.

CMPEQ(r0, r31, r1) compares 0 to 0 to see if they're equal. They are, so the result of the CMPEQ is 1, which is then stored in R1.

ADD(r1, r1, r2) adds 1 to 1, giving 2.

OR(r2, r1, r3) does a bit-wise logical OR of 2 (= 0b10) with 1 (= 0b01), giving 3 (= 0b11).

SHL(r2, r3, r4) shifts the value 3 to the left by 3 positions, i.e., 0b10 shift left by three positions, gives 0b10000 = 16.

SUB(r1, r2, r5) computes $1 - 2 = -1$

CMPLT(r31, r5, r6) is 0 less than or equal to -1? Nope, so the answer is 0. Note that the compare instructions treat the source operands as 32-bit two's complement numbers, which have negative values when their high-order bit is 1.

SHR(r5, r4, r7) does a logical right shift, which fills in the vacated bit positions with 0. So the value -1 = 0xFFFFFFFF shifted right by 16 bits gives 0x0000FFFF = 0xFFFF.

SRA(r5, r4, r8) does an arithmetic right shift, which fills in the vacated bit positions with sign bit from the A operand. So the value -1 = 0xFFFFFFFF arithmetically shifted right by 16 bits gives 0xFFFFFFFF = -1.

Assume that the first instruction is stored in location 0 of main memory.

What is the location for the CMPLT(r31, r5, r6) instruction?

0x0018

✓ Answer: 0x18

Explanation

Here's how the instructions are located in memory, one per 32-bit word:

location	instruction
0x00	ADD(r31, r31, r0)
0x04	CMPEQ(r0, r31, r1)
0x08	ADD(r1, r1, r2)
0x0C	OR(r2, r1, r3)
0x10	SHL(r2, r3, r4)
0x14	SUB(r1, r2, r5)
0x18	CMPLT(r31, r5, r6)
0x1C	SHR(r5, r4, r7)
0x20	SRA(r5, r4, r8)

Submit

❗ Answers are displayed within the problem

LE9.2.2: ALU Instruction Encoding

0.0/1.0 point (ungraded)

Please give the 32-bit binary encoding for the ALU instruction shown below. Start by figuring out the encoding for each of the 4 instruction fields, then concatenate the fields appropriately to determine the final 32-bit encoding.

MUL(R1, R17, R22)

You'll need the Summary of Instruction Formats handout to determine the encoding for the MUL opcode -- see the link in the first problem. As before, it's easiest to enter your answer in hex by using a "0x" prefix or binary by using a "0b" prefix.

Value of 6-bit opcode field?

Value of 5-bit RC field?

Value of 5-bit RA field?

Value of 5-bit RB field?

32-bit encoding for instruction?

LE9.2.3: ALU Instruction Encoding

0.0/1.0 point (ungraded)

Please give the 32-bit binary encoding for the ALU instruction shown below. Note that the symbolic form of our instructions allows as an operand any expression that yields a constant value! Only the low-order 5 bits of the value are used when filling in a 5-bit register field; the rest of the bits in the value are discarded.

SRA(R27, 3+0x11, -1)

Of course, we'd never write an instruction in so inscrutable a form. It would be impossible to read and understand :)

32-bit encoding for instruction?

Discussion

Topic: 9. Designing an Instruction Set / LE9.2**Add a Post**

Show all posts



by recent activity

Is the C program for factorial in the slide correct?

4






Hi, I have just started the course. In the very first lecture itself where we want to compute the fact...Program counter

2

What does the program counter do when it reaches the end? Am I misunderstanding something? T...Which one, signed or unsigned, is used in common in other architectures when CMP?

5

In the first problem asking for R6, I thought CMP would compare values in unsigned manner. But, I...

 <u>LE 9.2.1: How can I do the first problem without initial value of any register?</u> <u>How? I assumed that all registers are initialized as 0, but it is not correct.</u>	9
 <u>SUB(r1,r2,r5) computes 1-2=-1</u> <u>Please help me, LE9.2.1) according to \beta document SUB(r1,r2,r5) means $r1 \leftarrow (r2 - r5)$. However, r5...</u>	9
 <u>Chrome thinks Instruction Format .pdf is Chinese!</u>	5
<input checked="" type="checkbox"/> <u>wondering about instruction format choice?</u> <u>working through the lecture exercises and the beta documentation w/c I quote: "There are only tw...</u>	5
 <u>SRA</u> <u>Can someone explain meaning of a word **arithmetically** is a phrase "contents of register Ra are...</u>	3
 <u>SHR(r5,r4,r7) does what?</u> <u>"The contents of register Ra are shifted right 0 to 31 bits as specified by the five-bit count in regist...</u>	2