LE9.5

Bookmark this page

LE9.5.1: Branch Instructions

1.0/1.0 point (ungraded)

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

Consider the execution of a short program that loops to sum the elements of an array with 4 elements. The first element of the array is stored at location 0×2000.

```
  . = 0                 // first instruction is at location 0
  ADDC(r31,array,r0)    // r0 = pointer to next array element
  ADDC(r31,4,r1)        // r1 = number of array elements remaining
  ADDC(r31,0,r2)        // r2 = accumulated sum
loop:
  LD(r0,0,r3)           // load next value from array
  ADD(r3,r2,r2)         // add to sum
  ADDC(r0,4,r0)         // increment pointer to next word
  SUBC(r1,1,r1)         // decrement counter
  BNE(r1,loop,r31)      // loop if more elements to go
  ST(r2,result,R31)     // write result to memory
  // execution stops here

  . = 0x2000
array:
  LONG(1)               // array[0] = 1
  LONG(2)               // array[1] = 2
  LONG(3)               // array[2] = 3
  LONG(4)               // array[3] = 4
result:
  LONG(0)               // where result will be stored
```

Program execution starts with the first instruction and halts after execution of the ST instruction.

(A) What value does the assembler give the label "loop"? 0×0c ✔

(B) After execution, number of times `LD` is executed? 4 ✔

(C) After execution, value left in r0? 0×2010 ✔

(D) After execution, value left in r1? 0 ✔

(E) After execution, value left in r2? 0×0a ✔

(F) After execution, value left in r3? 0×04 ✔

The encoding for the OPCODE, RC and RA fields of a branch instruction is just like the encodings for other Beta instructions. Figuring out the value for the 16-bit constant field takes a little more work. The offset value is the number of words between the instruction following the branch (ST in this example) to target instruction (LD in this example). Positive values indicate a forward branch to a subsequent location with a higher address; negative offset values indicate a backward branch to a location with a lower address.

In this example, we'd start counting instructions backwards from the store instruction until we reached the LD instruction. Since it's a backwards branch, we'd encode the count as a negative number in the 16-bit constant field of the BNE instruction.

(G) What is the binary encoding for `BNE(r1,loop,r31)`? 0b01110111111000011111′ ✔

Submit

Calculator

Discussion

Topic: 9. Designing an Instruction Set / LE9.5

Hide Discussion

Add a Post

Show all posts by recent activity

?	Effective Address and sign-extended 16-bit displacement literal? When I try to understand what is going on behind in LD and ST instruction, I could not understand exactly these two definition. What ...	3
☑	label or literal? Why is it that in the Beta Documentation, we sometimes use **label**, and sometimes **literal**? Example: **BNE/BT(Ra,label,Rc): R...	6
💬	LE9.5G opcode [staff] I believe the answer for LE9.5 part G incorrectly codes the BEQ opcode rather than the BNE opcode.	3
💬	sign extension for RC? [STAFF] Just to clarify, for part G is the binary encoding for RC 11111 because the program counter address 32 (100000) was sign extended t...	2

Previous Next