**Video explanation of solution is provided below the problem.**

**For all Beta related questions, you should make use of the Beta documentation, the Beta Instruction Summary, and the Beta Diagram.**

## Beta Assembly

7/7 points (ungraded)

For each of the Beta instruction sequences shown below, indicate the values of the specified quantities after the sequence has been executed. Consider each sequence separately and assume that execution begins at location 0 and halts when the HALT() instruction is about to be executed. Also assume that all registers have been initialized to 0 before execution begins. Remember that even though the Beta loads and stores 32-bit words from memory, all addresses are *byte addresses*, i.e., the addresses of successive words in memory differ by 4.

Fill in requested values left after execution of each segment, or “CAN'T TELL” where appropriate.

1.

```
     . = 0
     LD(R31, c, R0)
     ADDC(R0, b, R0)

     HALT()

     . = 0x200
a:   LONG(0x100)
b:   LONG(0x200)
c:   LONG(0x300)
```

**Value left in R0 (HEX): 0x**

504

✓ Answer: 504

**Value assembler assigns to the symbol “c”: 0x**

208

✓ Answer: 208

Explanation

The contents of label **c** are first loaded into R0, and then the value of **b** is added to it. Since we are told that a is at location 0×200, that means that the value of b is 0×204. So the value left in R0 = 0×300 + 0×204 = 0×504.
Just as b was 0×204, c is one word after that which is 0×208.

2.

```
      . = 0
      BR(. + 4, R0)
      HALT()
```

**Value left in R0: 0x**

004

✓ Answer: 4

Explanation
The **BR** instruction branches to address 4 which is the address of the HALT() instruction, and stores the address of the instruction immediately following the **BR** into R0. So R0 = 4.

3.

```
      . = 0
      LD(R31, x, R0)
      CMOVE(0, R1)

loop: ANDC(R0, 1, R3)
      ADD(R3, R1, R1)
      SHRC(R0, 1, R0)
      BNE(R0, loop)
      HALT()

x:    LONG(0x0FACE0FF)
```

**Value left in R0: 0x**

0

✓ Answer: 0

**Value left in R1: 0x**

13

✓ Answer: 13

Explanation

This code counts the number of 1's in the value loaded into R0 which is 0×0FACE0FF = 0b00001111101011001110000011111111. There are 19 1's in this number which is 0×13. The loop halts when all the 1's in R0 have been shifted out and R0 = 0.

4.
```
    . = 0
    CMOVE(0x1000, SP)
    PUSH(SP)

    HALT()
```

**Value left in SP (HEX): 0x**

1004 ✓ Answer: 1004

**Value pushed onto stack (HEX): 0x**

1004 ✓ Answer: 1004

Explanation

The CMOVE instruction makes SP = 0×1000 and the PUSH(SP) increments the SP by 4 so SP = 0×1004.

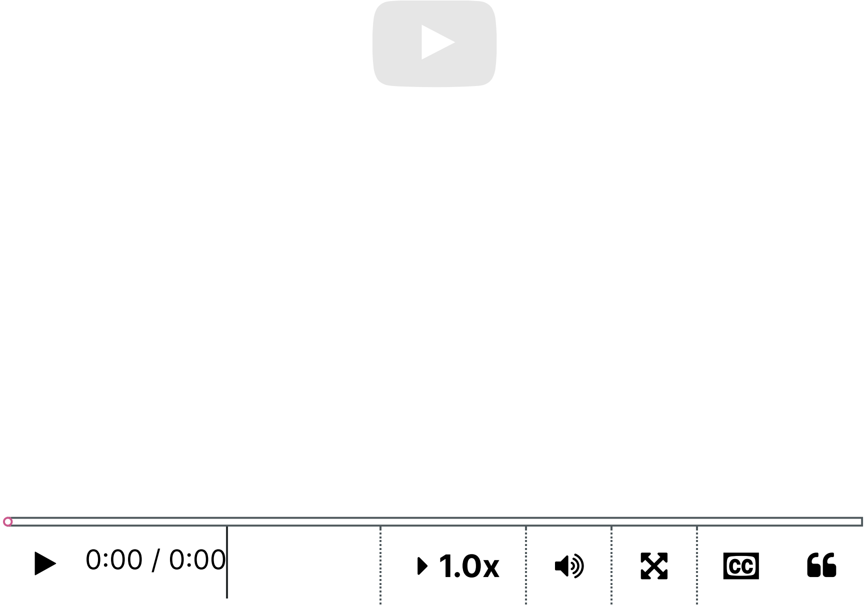
Since the PUSH macro first increments the SP and then stores the value being pushed into SP-4, the value pushed onto the stack is 0×1004.

Submit

Answers are displayed within the problem

## Beta Assembly

Start of transcript. Skip to the end.



As presented in lecture, in this course, we use a simple 32-bit processor called the Beta.

The Beta works on 32-bit instruction and data words.

However, the addresses in memory are specified in bytes.

### Video

Download video file

### Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

## Discussion

Hide Discussion

**Topic:** 10. Assembly Language, Models of Computation / WE10.1

**Add a Post**

Show all posts by recent activity

- ☑ BEQ definition - edit? 4
- ☑ WE10 - Runtime error when attempting 'PUSH(SP)' 7
  When I try to run these instructions in BSim: . = 0 CMOVE(0×1000, SP) PUSH(SP) HALT() I get the f...
- ☑ Words layout in memory. 2
  Hi. Suppose we have following instruction > x: LONG(0×0FACE0FF) It's memory layout (Little Endia...
- 💬 Value Left in R0 3
  I'm confused with the first conclusion that R0 contains 0×300 after the first instruction. My unders...
- ☑ Where did this 8 come from? 8