计算机体系结构 Exercise 03

邱一航, Yihang Qiu, 2021/10/13

5.9. Solution:

(a) The instruction can be interpreted as "`R1←R1+0`".

However, since the instruction is an ADD instruction, the value of condition codes `n/z/p` might be changed. Thus, it is not an NOP.

(b) The instruction can be interpreted as "`If True, PC←PC+1`", **i.e.** "`PC←PC+1`".

However, considering PC has already been incremented at this time, the instruction acts as "ignoring the following instruction".

(c) The instruction can be interpreted as "`If False, PC←PC+0`", **i.e.** NOP.

In conclusion, **only (c)** can act as NOP instructions. ■

The difference among (a) and other two instructions:

1) (a) uses ALU while the other two does not.

2) (a) changes the value of condition codes `(n/z/p)` while the other two does not.

3) (a) fetches value (operand) from register file while the other two fetch value (operand) from PC.

■

5.16. Solution:

(a) PC-related Mode (LD) is the best choice. ■

Explanations:

The offset in PC-related Mode is of 9 bits and is thus able to reach the range of $\pm 2^8$ locations away. Using Indirect Mode and Base + Offset Mode is acceptable but more complicated and time-consuming.

(b) Indirect Mode (LDI) and **Base + Offset Mode (LDR)** are both best choices. ■

Explanations:

In this case, the range PC-related Mode can visit is smaller than required.

Indirect Mode can load one value from the location whose address is stored in the Memory. Base + Offset Mode can load one value from a location whose address is several locations away from an address stored in the register.

(c) Base + Offset Mode (LEA+LDR) is the best choice. ■

Explanations:

We can get the address of the first location in the array by LEA and load it into a register. Using Base + Offset Mode (LDR instruction) can easily load value from the whole array by incrementing the offset.

5.22. Solution:

`PC←x3011` **Instruction:** "`R3←PC+x3F`". Thus, R3 = x3050.

`PC←x3012` **Instruction:** "`R4←[R3+x00]`". Thus, R4 = [x3050] = x70A4.

`PC←x3012` **Instruction:** "`R6←[R4+x00]`". Thus, R6 = [x70A4] = x123B.

Therefore, the value to be loaded into R6 is **x123B**. ■

We could replace the three instructions above with the following instruction:

```
1010 1100 0011 1111
```
■

5.23. Solution:

`PC←x3100` **Instruction:** "`R1←PC+x01`". Thus, R3 = x3101.

`PC←x3101` **Instruction:** "`R2←[R1+x02]`". Thus, R2 = x1482.

`PC←x3102` **Instruction:** "`TRAP x25`". Halt.

Thus, the value of R2 after the execution of program is **x1482**. ■

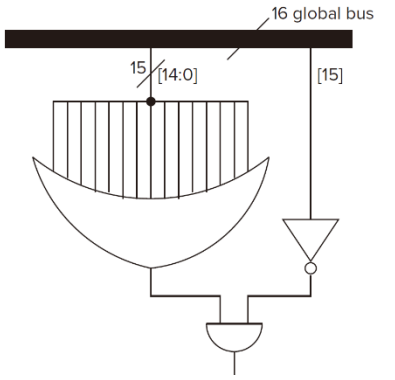
5.40. Solution:

The opcode of the circuit is 0000, **i.e.** BR instruction. Only when one of the condition code N, Z, P is examined set will the output of A be 1.

Thus, the signal A tells **whether the condition given by the instruction is reached or not**. (If the condition is reached, A=1; otherwise, A=0.)

■

5.41.(a) Solution:



Since the 15th bit of the datum from the global bus is the sign bit and the 15-input OR Gate only produces a "0" when all 15 inputs are 0, we know the output of the right upper part of the circuit (depicted in the figure on the right side) tells whether the value from the global bus is positive or not.

The right lower part of the circuit is a D-Latch.

Therefore, signal Y provides **the information of "p" condition code**. ■

(b) Solution:

The content of "p" decides on the result loaded into the general-purpose registers by ADD, AND, NOT, LD, LDI, LDR and LEA instructions. Meanwhile, opcodes related in the circuit are 0000(BR), 0101(AND), 0010(LD), 1010(LDI), 0110(LDR), 1110(LEA), 1001(NOT). However, the BR instruction shall not change the value of "p" condition code.

In other words, there exists an error in the logic generating signal X, **i.e. allowing BR instruction to alter the value of "p" condition code while prohibiting ADD instruction to alter the value of "p" condition code.** ■