Condition Logic

[Introduction](#)

[History](#)

Design

[Circuit Design](#)
[Architecture](#)
[Conditional Logic](#)
[Semiconductors](#)
[Instruction Set](#)

Usage

[Keypad/Display](#)
[Serial Console](#)
[Example Programs](#)

[Software Tools](#)
[Build/Dev Log](#)

[Project page](#)
[Comments](#)

Condition Logic

The condition logic supports two types of tests: comparisons with zero and comparisons between two unsigned numbers.

A trick is used to detect if a value is equal to zero. Instead of having dedicated zero-detect logic (which would normally be implemented as a input NOR gate), the carry chain of the ALU is used. If the value under test is negated, then the carry output from the ALU will be high if an if the input value is exactly zero.

Comparisons with zero

In this mode, the A field will indicate a memory location to test and the B field will specify a jump address to take if the condition is true. Th ALU should be set up to negate the A operand in order to test for zero. If the A operand is zero, the carry out from the ALU will be 1.

Instruction set up for comparison with zero:

- CEN = 0
- CINV = 1
- COM = 1
- BEN = 0
- A has address of data to test
- B has jump address

Comparison between two unsigned arguments

First, a subtraction instruction is issued with the two arguments to compare. In other words, compute X - Y. The result should be saved back t order for the second instruction to test it for zero.

Instruction set up for subtraction part:

- B points to the left side argument X
- A points to the right side argument Y
- CEN = 0
- CINV = 1
- COM = 1
- BEN = 1
- WRB = 1

Second, issue the conditional instruction with the subtraction result as the A operand to the ALU.

Instruction setup for conditional jump part:

- A points to X (which was written to by the previous instruction).
- B has the jump address.
- CEN = 0
- CINV = 1
- COM = 1
- BEN = 0

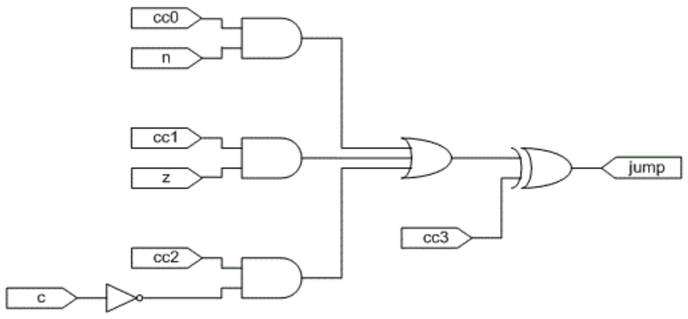
Condition Codes

Here is a summary of the condition codes. Key:

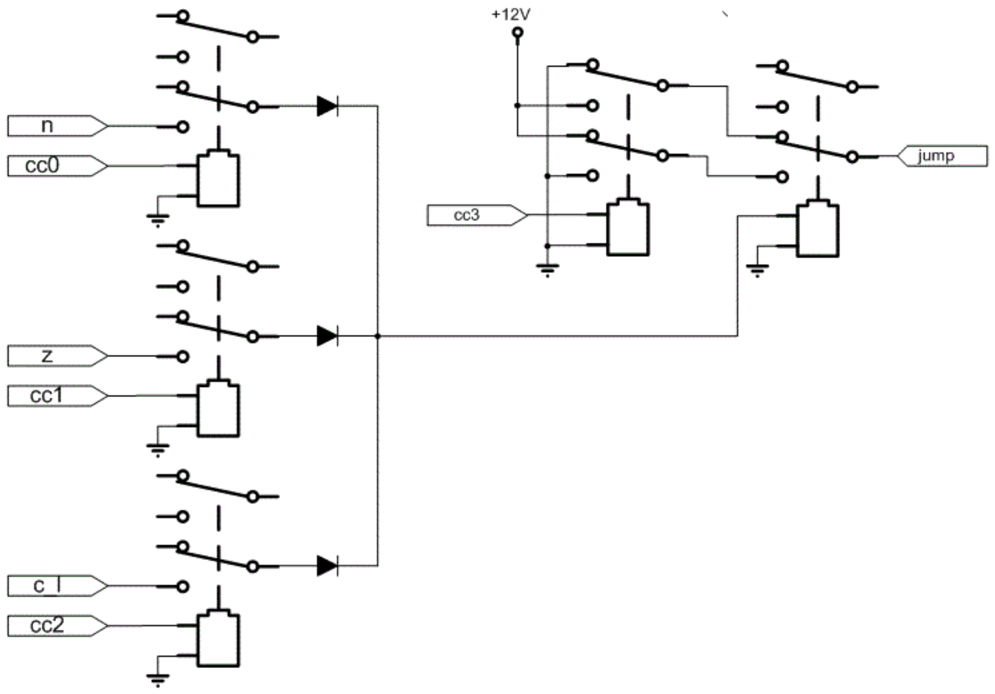
- N is bit 7 of the A argument of the ALU.
- Z is the carry-out value of the ALU (which is the input to the carry flag).
- C is the carry flag (the carry-out of the ALU from the previous instruction).
- A is A operand to the ALU
- X and Y are arguments to an immediately preceding subtraction instruction (performing X - Y). The result of this instruction should be presented to the A argument of the ALU for the zero test.

Code	Jump if		Code	Jump if	
0	0	No jump	8	1	Jump
1	N==1	A<0	9	N==0	A>=0
2	Z==1	A==0	A	Z==0	A!=0
3	N==1 || Z == 1	A<=0	B	N==0 && Z==0	A>0
4	C==0	X<Y	C	C==1	X>=Y
5	C==0 || N==1		D	C==1 && N==0	
6	C==0 || Z == 1	X<=Y	E	C==1 && Z==0	X>Y
7	C==0 || Z==1 || N==1		F	C==1 && N==0 && Z==0	

The logical design of the condition decoder is as follows. cc0 - cc3 are the 4 bits from the cc field of the instruction.



It is implemented with 5 relays as follows. The carry flag provides both inverting and non-inverting outputs, so c_l comes directly from it.


n
cc0
z
cc1
c_l
cc2
+12V
cc3
jump