

L4 (CHAPTER 7)

Programming in Assembly Part 3: Control Structures Exercises

Condition Codes

Condition Code	Meaning	Requirements
EQ	Equal	$Z = 1$
NE	Not equal	$Z = 0$
CS	Carry set	$C = 1$
CC	Carry clear	$C = 0$
MI	Minus/negative	$N = 1$
PL	Plus/positive or zero (non-negative)	$N = 0$
VS	Overflow	$V = 1$
VC	No overflow	$V = 0$
HI	Unsigned $>$ ("Higher")	$C = 1 \ \&\& \ Z = 0$
LS	Unsigned \leq ("Lower or Same")	$C = 0 \ \ Z = 1$
GE	Signed \geq ("Greater than or Equal")	$N = V$
LT	Signed $<$ ("Less Than")	$N \neq V$
GT	Signed $>$ ("Greater Than")	$Z = 0 \ \&\& \ N = V$
LE	Signed \leq ("Less than or Equal")	$Z = 1 \ \ N \neq V$
AL	Always (unconditional)	only used with IT instruction

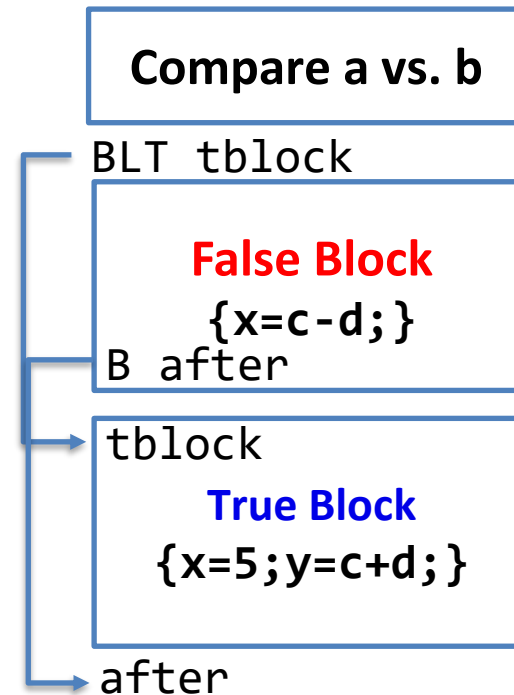
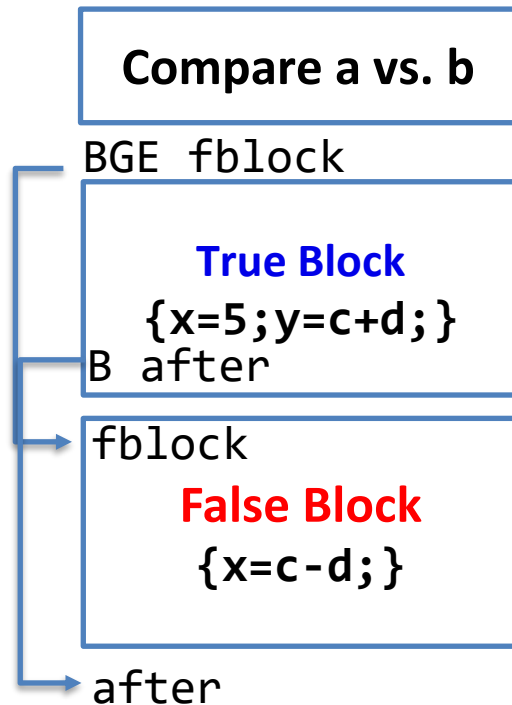
The condition is described as the state of a specific bit in the CPSR register. For example, when we compare two numbers a and b , and they turn out to be equal, we set the Zero bit ($Z = 1$), because $a - b = 0$. In this case we have EQual condition. If the first number was bigger, we would have a Greater Than condition and in the opposite case – Lower Than. There are more conditions, like Lower or Equal (LE), Greater or Equal (GE) and so on. Any one of these may be appended to any instruction mnemonic when used inside an If-Then-Else (IT) block.

Another Example:

- C code:

```
if (a < b) {x = 5; y = c + d;} else {x = c - d;}
```

- Assembler code:



Question: Conditional

- C program:
`if (a == 0) { //True Block } else { //False Block }`
- Write the assembler program for the C program, given the snippets provided

```
ADR R4,a
LDR R0,[R4]
_____
_____
tblock % True Block
_____
fblock % False Block
after
```

```
ADR R4,a
LDR R0,[R4]
_____
_____
fblock % False Block
_____
tblock % True Block
after
```

Loops: Predetermined #Iterations

C code:

```
for (n = 0; n < 100; n++)  
{  
    ... //Loop body  
}
```

Assembler code (option 1):

```
        LDR    R0,=0  
top:    CMP    R0,#100  
        BGE    done ;Branch  
greater than or equal to (n>=100)  
        ...  
        ADD    R0,R0,#1  
        B      top  
done:
```

Assembler code (option 2):

```
        LDR    R0,=0  
top:    ...  
        ADD    R0,R0,#1  
        CMP    R0,#100  
        BLT    top ;Branch less  
than (n<100)  
done:
```

More efficient, with fewer
branch instructions.

Question: Loop

- Q: How many iterations does the following loop execute?
 - (a) for ($n=0$; $n<100$; $n += 2$) {...}
 - (b) for ($n=0$; $n<100$; $n *= 2$) {...}
 - (c) for ($n=1$; $n<100$; $n *= 2$) {...}

Question: Loop

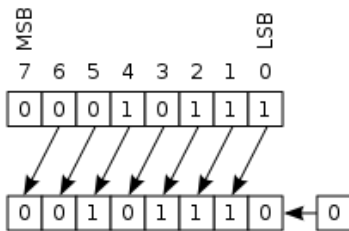
- Write the assembler code for the following C program

For (n=1; n<100; n *=2) {...}

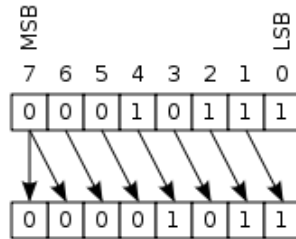
Review

Shift Instructions

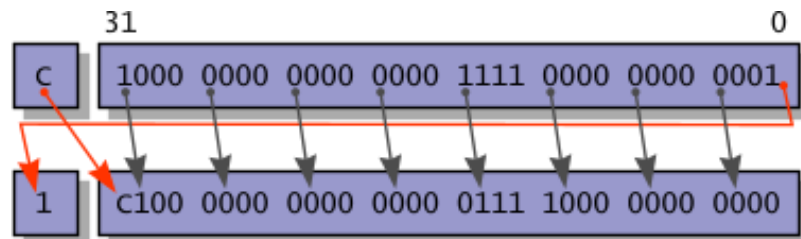
<shift>	Meaning	Notes
LSL #n	Logical shift left by n bits	Zero fills; $0 \leq n \leq 31$
LSR #n	Logical shift right by n bits	Zero fills; $1 \leq n \leq 32$
ASR #n	Arithmetic shift right by n bits	Sign extends; $1 \leq n \leq 32$
ROR #n	Rotate right by n bits	$1 \leq n \leq 32$
RRX	Rotate right w/C by 1 bit	including C bit from CPSR



LSL #1



LSR #1



RRX

Any of these may be applied to the 2nd operand register in Move / Add / Subtract, Compare, and Bitwise Groups.