Part 0x6

CHAPTER 3

Architecture and Organization
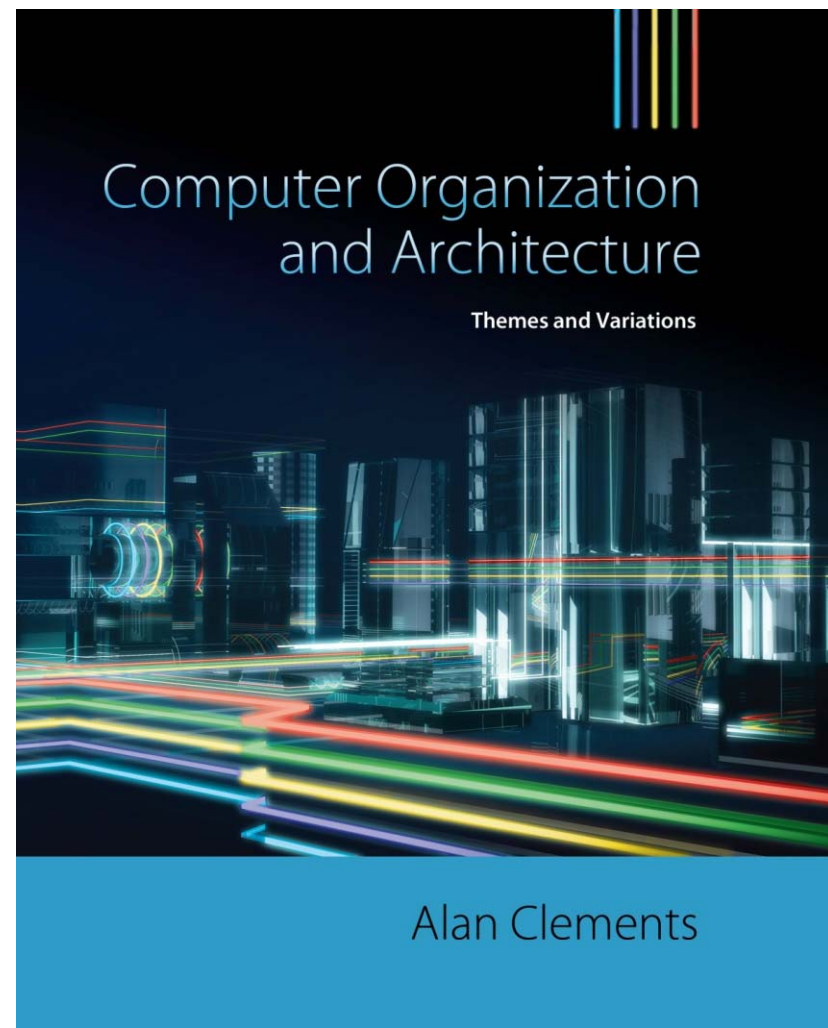
Computer Organization and Architecture

Themes and Variations

Alan Clements

1

Music: *"Corporate Success"* by *Scott Holmes*, used under *Attribution-NonCommercial License*

CENGAGE Learning™

ARM's Flow Control Instructions
(Unconditional Branch)

❑ **ARM**'s ***unconditional branch*** *instruction* has the form `B target`, where `target` denotes the *branch target address* which is *the address of the next instruction to be executed*.

❑ The following fragment of code demonstrates how the unconditional branch is used.

```
..      do this       ;Some code
..      then that      ;Some other code
        B   Next       ;Now skip past next instructions
..                     ;…the code being skipped
..                     ;…the code being skipped
Next ..                ;Target address for the branch
```

❑ In a high-level language, the unconditional branch is called a *goto*, which is considered a poor programming style;

❑ *Yet, in assembly, the unconditional branching is unavoidable*,
   o Assembly is a low-level language which ***does not*** have
      built-in constructs such as *if …then.. else*, *while*, *repeat*, *for*, …

80

ARM's Flow Control Instructions (Conditional Branch)

❑ Consider the following if statement,
   IF (X == Y)
   THEN Y = Y + 1
   ELSE Y = Y + 2

❑ A test is performed, and one of the two courses of action is carried out depending on the outcome.
❑ We can translate this as:

```
        CMP r1,r2     ;Compare r1 and r2,
                      ;where r1 contains y and r2 contains x
        BNE Plus2     ;if not equal then branch to the else part
        ADD r1,r1,#1 ;if equal fall through to here
                      ;and add one to y
        B   leave     ;now skip past the else part
Plus2 ADD r1,r1,#2 ;ELSE part add 2 to y
leave   …             ;continue from here
```

81

ARM's Flow Control Instructions
(Conditional Branch)

❑ The *conditional branch* *instruction*
- o tests the flag bits (*condition codes*) in the *current program status register* (*CPSR*), then
- o takes the branch if the tested condition is true.

❑ **ARM** dedicates 4 bits in each instruction to encode *16 different conditions* in total
- o **eight** possible conditional branches based on the state of a *single bit*, namely Zero bit (Z), Negative bit (N), Carry bit (C), and oVerflow bit (V):
  - ▪ *four* that *branch on **true*** and
  - ▪ *four* that *branch on **false***.
- o **six** compound conditional branches
- o **one** always branch (unconditional)
- o **one** never branch (reserved)

82

ARM's Flow Control Instructions (Conditional Branch)

TABLE 3.2	ARM's Conditional Execution and Branch Control Mnemonics		
Encoding	**Mnemonic**	**Branch on Flag Status**	**Execute on condition**
0000	EQ	Z set	Equal (i.e., zero)
0001	NE	Z clear	Not equal (i.e., not zero)
0010	CS	C set	Unsigned higher or same
0011	CC	C clear	Unsigned lower
0100	MI	N set	Negative
0101	PL	N clear	Positive or zero
0110	VS	V set	Overflow
0111	VC	V clear	No overflow
1000	HI	C set and Z clear	Unsigned higher
1001	LS	C clear or Z set	Unsigned lower or same
1010	GE	N set and V set, or N clear and V clear	Greater or equal
1011	LT	N set and V clear, or N clear and V set	Less than
1100	GT	Z clear, and either N set and V set, or N clear and V clear	Greater than
1101	LE	Z set, or N set and V clear, or N clear and V set	Less than or equal
1110	AL		Always (default)
1111	NV		Never (reserved)

83

© Cengage Learning 2014

ARM's Flow Control Instructions
(Branching and Loop Constructs)

❑ Nothing illustrates the concept of flow control better than the classic loop constructs that are at the core of so-called structured programming.

❑ The following demonstrate the structure of

    ❑ WHILE loop,

    ❑ REPEAT-UNTIL loop, and

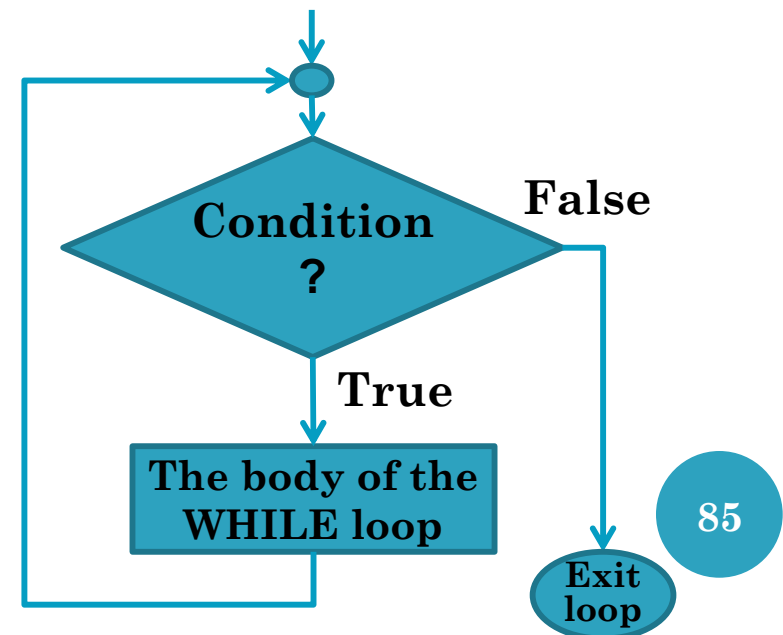   ❑ FOR loop

84

ARM's Flow Control Instructions (Branching and Loop Constructs)

The WHILE loop example

```
While       CMP    r0,#0        ;perform test at start of loop
            BNE    Exit         ;exit

            code  ...           ;body of the loop

            B      While        ;loop again WHILE true
Exit        Post-loop ...       ;Exit
```

WHILE(r0 == 0)
{ code;
}

WHILE(condition is TRUE)
{ The body of the WHILE loop;
}

ARM's Flow Control Instructions
(Branching and Loop Constructs)

The REPEAT-UNTIL loop example

```
Repeat      code  ...              ;body of the loop


            CMP   r0,#0            ;perform test at end of loop
            BNE   Repeat           ;loop again UNTIL true
Exit        Post-loop ...          ;Exit
```
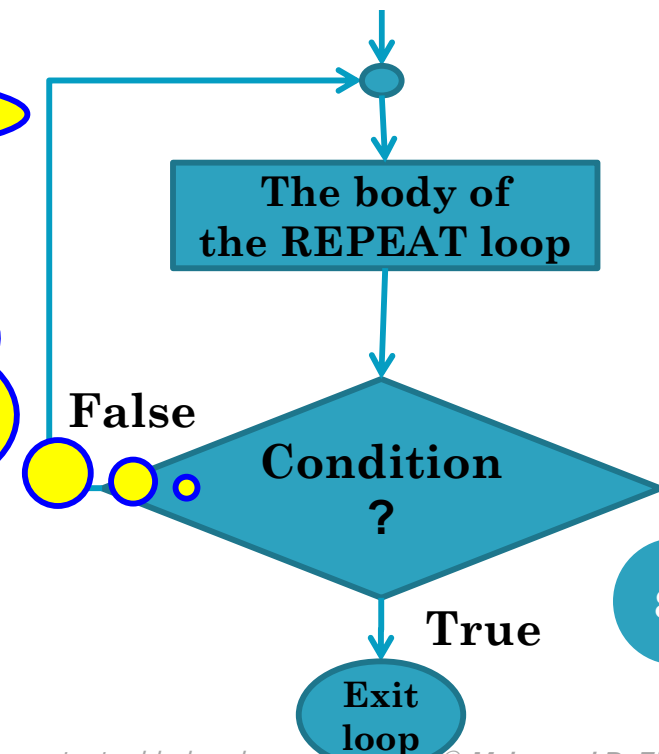
C and Java implement this loop using
DO {code} WHILE (condition is TRUE)

REPEAT
{ code;
} UNTIL (r0 == 0)

In C and Java, the
looping occurs when
the condition is true
and exiting the loop
when it is false.

REPEAT
{ The body of the REPEAT loop;
} UNTIL (condition is TRUE)

The body of
the REPEAT loop

False

Condition
?

True

Exit
loop

86

ARM's Flow Control Instructions
(Branching and Loop Constructs)

The FOR loop example

```
       MOV    r0,#10       ;set up the loop counter

Loop code   ...            ;body of the loop

       SUBS   r0,r0,#1     ;decrement loop counter,
                           ;set flags
       BNE    Loop         ;continue until
                           ;count zero
       Post   loop ...     ;fall through on
                           ;zero count
```
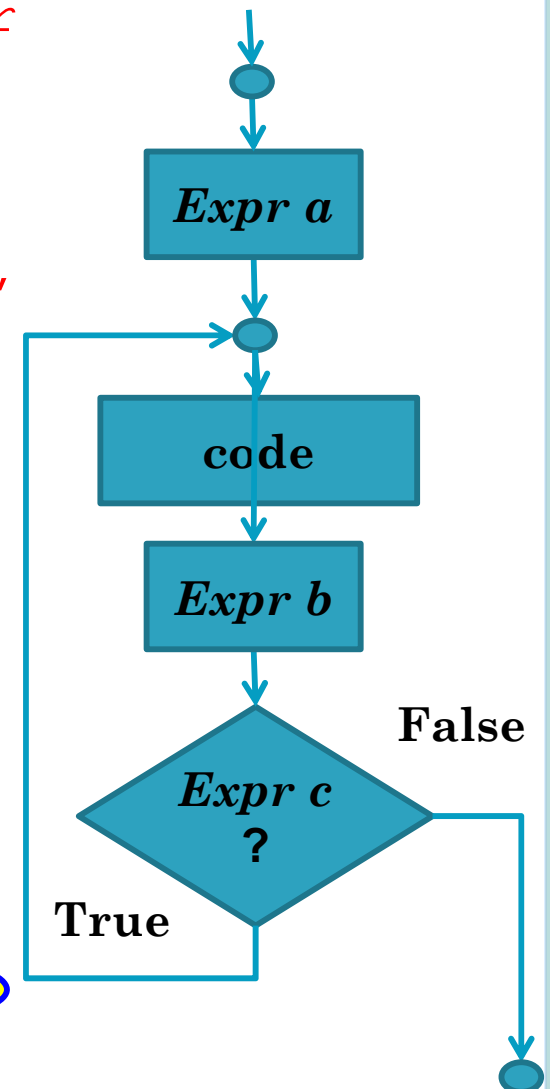
This FOR loop is different than the C and Java FOR loop.

The C and Java FOR loop has "Expr c" at the beginning of the loop, not at the end of it.

Expr a

Expr b

code

Expr c ?

False

True

ARM's Flow Control Instructions
(Branching and Loop Constructs)

The combination loop example

```
        MOV   r0,#10        ;set up the loop counter
LoopStart CMP   r1,#0         ;perform test at start of loop
        BNE   ComboExit     ;exit on test true

        code  ...           ;body of the loop

        CMP   r2,#0         ;perform test at end of loop
        BEQ   ComboExit     ;exit on test true

        SUBS  r0,r0,#1      ;decrement loop counter, set flags
        BNE   LoopStart     ;continue until count zero
ComboExit Post  loop ...     ;Exit
```

88