Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

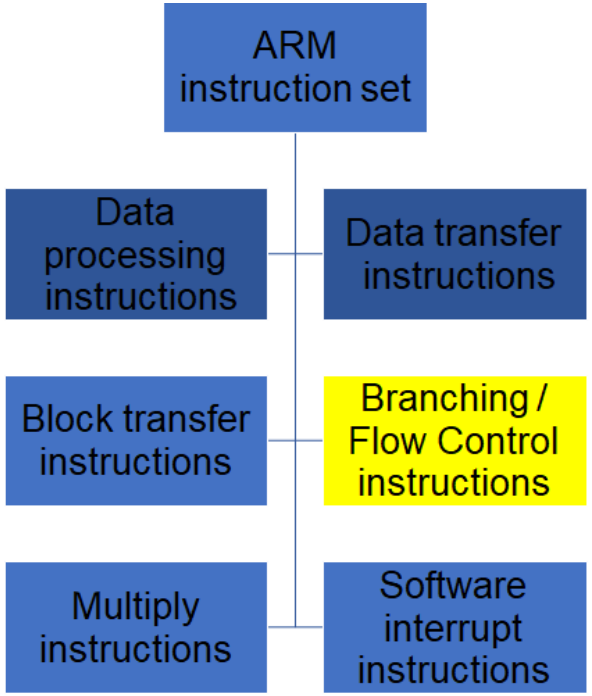
Flow Control Instructions

Dr. D. C. Kiran
Department of Computer Science and Engineering

Syllabus

Unit 1: Basic Processor Architecture and Design

- ~~Microprocessor Overview~~
- ~~CISC VS RISC~~
- ~~Introduction to ARM Processor & Applications~~
- ~~ARM Architecture Overview~~
- ~~Different ARM processor Modes~~
- ~~Register Bank~~
- ~~ARM Program structure~~
- ~~ARM Instruction Format~~
- **ARM INSTRUCTION SET**
  - ~~Data Processing Instructions~~
  - Branch Instructions



Flow Control Instructions

Syntax: B{<cond>} Label
BL{<cond>} Label
BX{<cond>} Rm
BLX{<cond>} Rm

B	Branch	Program Counter = Label
BL	Branch & Link	**Step1:** PC will be copied to R14 the Link Register (LR) before branch is taken. **Step2:** Program Counter = Label
BX	Branch Exchange	Used for changing ARM to Thumb mode or from Thumb mode to ARM mode. Reference
BLX	Branch Exchange with link	

Flow Control Instructions

- **Branch instruction**

```
        B   label
        …
label:  …
```

- **Conditional branches**

```
        MOV   R0, #0
loop:         …
        ADD   R0, R0, #1
        CMP   R0, #10
        BNE   loop
```

Branch and Link

- **BL** instruction saves the return address to **R14** (lr)

```
      BL     sub       @ call sub
      CMP    R1, #5    @ return to here
      MOVEQ  R1, #0
      …
sub:  …                @ sub entry point
      …
      MOV    PC, LR    @ return
```

Branch and Link (Nested Procedure Call)

```
        BL      sub1        @ call sub1
```

use stack to save/restore the return address and registers

```
sub1:   STMFD R13!, {R0-R2,R14}
        BL      sub2
        …
        LDMFD R13!, {R0-R2,PC}

sub2:   …
        MOV     PC, LR
```

Conditional Branch Options

Branch	Interpretation	Normal uses
B BAL	Unconditional Always	Always take this branch Always take this branch
BEQ	Equal	Comparison equal or zero result
BNE	Not equal	Comparison not equal or non-zero result
BPL	Plus	Result positive or zero
BMI	Minus	Result minus or negative
BCC BLO	Carry clear Lower	Arithmetic operation did not give carry-out Unsigned comparison gave lower
BCS BHS	Carry set Higher or same	Arithmetic operation gave carry-out Unsigned comparison gave higher or same
BVC	Overflow clear	Signed integer operation; no overflow occurred
BVS	Overflow set	Signed integer operation; overflow occurred
BGT	Greater than	Signed integer comparison gave greater than
BGE	Greater or equal	Signed integer comparison gave greater or equal
BLT	Less than	Signed integer comparison gave less than
BLE	Less or equal	Signed integer comparison gave less than or equal
BHI	Higher	Unsigned comparison gave higher
BLS	Lower or same	Unsigned comparison gave lower or same

Conditional Branch Example1

Compare the value of R0 and R1, add if R0 = R1, else subtract

```
.text
        MOV r0, #5
        MOV r1, #10
        CMP r0, r1
        BEQ L1 // Conditional Branch Equal
        SUB r2, r1, r0
        B L2        // Unconditional Branch
        L1: ADD r2, r0, r1
        L2: SWI 0x011
.end
```

```
R0        :5
R1        :10
R2        :5
R3        :0
R4        :0
R5        :0
R6        :0
R7        :0
R8        :0
R9        :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4124
------------------
CPSR Register
Negative(N):1
Zero(Z)     :0
Carry(C)    :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
------------------
```

Compare the value of R0 and R1, add if R0 = R1, else subtract

```
.text
        MOV r0, #10
        MOV r1, #10
        CMP r0, r1
        BEQ L1
        SUB r2, r1, r0
        B L2
        L1: ADD r2, r0, r1
        L2: SWI 0x011
.end
```

```
R0        :10
R1        :10
R2        :20
R3        :0
R4        :0
R5        :0
R6        :0
R7        :0
R8        :0
R9        :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4124
------------------
CPSR Register
Negative(N):0
Zero(Z)     :1
Carry(C)    :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
```

;Based on the value of the number in R0 :
;Store 1 in R1 if R0 is zero
;Store 2 in R1 if R0 is positive
;Store 3 in R1 if R0 is negative

Mov R0, #10	R0 has the value 10
Cmp R0, #0	10!=0 . Update CPSR.z =0
Moveq R1, #1	Not Executed since CPSR.z=0
Beq L1	Branch not taken since CPSR.z=0
Movmi R1, #3	Not Executed, since CPSR.n=0
Bmi L1	Branch Not taken, since CPSR.n=0
Mov R1, #2	R1=2
L1:	
Swi 0x1011	

;Based on the value of the number in R0 :

;Store 1 in R1 if R0 is zero

;Store 2 in R1 if R0 is positive

;Store 3 in R1 if R0 is negative

Mov R0, #-10	R0 has the value -10, CPSR.n=1
Cmp R0, #0	-10!=0 . Update CPSR.z =0
Moveq R1, #1	Not Executed since CPSR.z=0
Beq L1	Branch not taken since CPSR.z=0
Movmi R1, #3	Executed, since CPSR.n=1, R1=3
Bmi L1	Branch taken, since CPSR.n=1. Jump to L1
Mov R1, #2	Control will not reach this instruction
L1:	
Swi 0x1011	

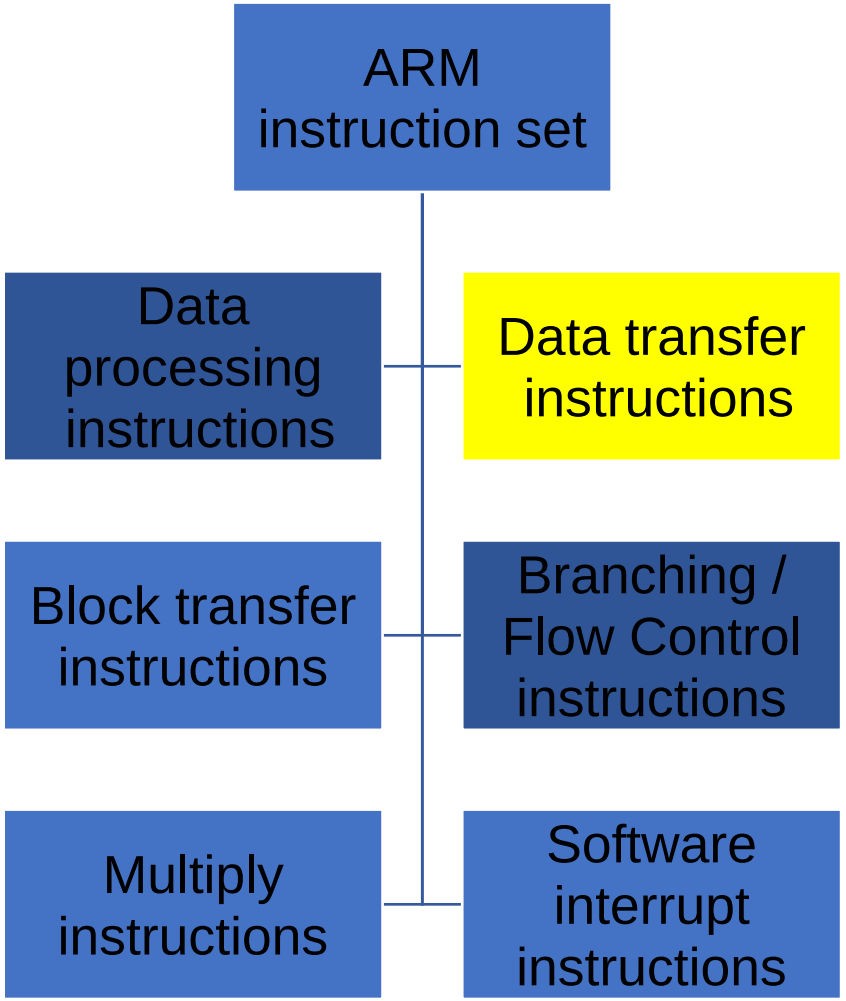
;Based on the value of the number in R0 :
;Store 1 in R1 if R0 is zero
;Store 2 in R1 if R0 is positive
;Store 3 in R1 if R0 is negative

Mov R0, #0	R0 has the value 0
Cmp R0, #0	0==0 . Update CPSR.z =1
Moveq R1, #1	Executed since CPSR.z=1, R1=1
Beq L1	Branch taken since CPSR.z=1
Movmi R1, #3	Control will not reach this instruction
Bmi L1	Control will not reach this instruction
Mov R1, #2	Control will not reach this instruction
L1:	
Swi 0x1011	

Looping

;Factorial of a given number

```
mov r0, #5
mov r1, #1
l1:         mul r1, r0, r1
            subs r0, r0, #1
bne l1                          ;Comparison not equal or non-zero results
```


ARM instruction set
Data processing instructions
Data transfer instructions
Block transfer instructions
Branching / Flow Control instructions
Multiply instructions
Software interrupt instructions

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135