



Microprocessor & Computer Architecture (μ pCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μ pCA)

Data Processing Instructions

Dr. D. C. Kiran

Department of Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Syllabus



Unit 1: Basic Processor Architecture and Design

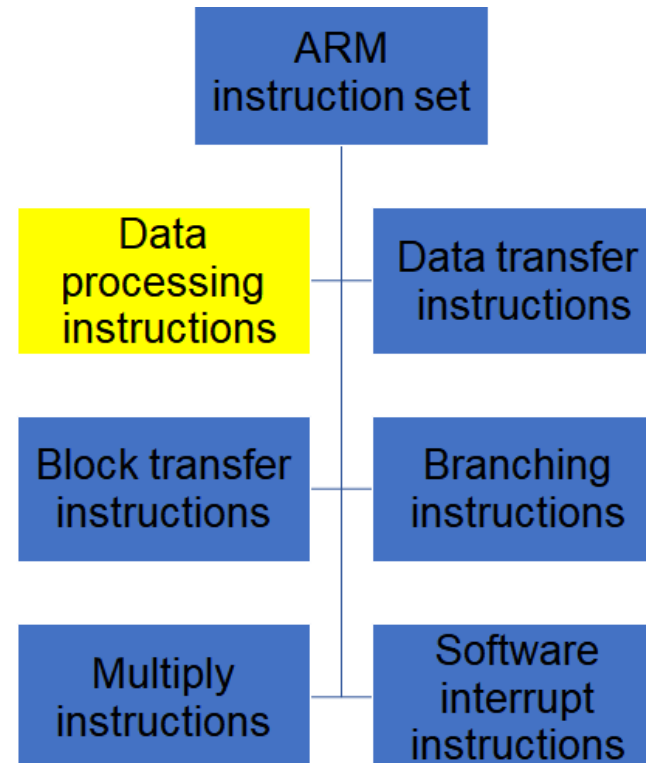
- ~~Microprocessor Overview~~
- ~~CISC VS RISC~~
- ~~Introduction to ARM Processor & Applications~~
- ~~ARM Architecture Overview~~
- ~~Different ARM processor Modes~~
- ~~Register Bank~~
- ~~ARM Program structure~~
- ~~ARM Instruction Format~~
- **ARM INSTRUCTION SET**

Data Processing Instructions

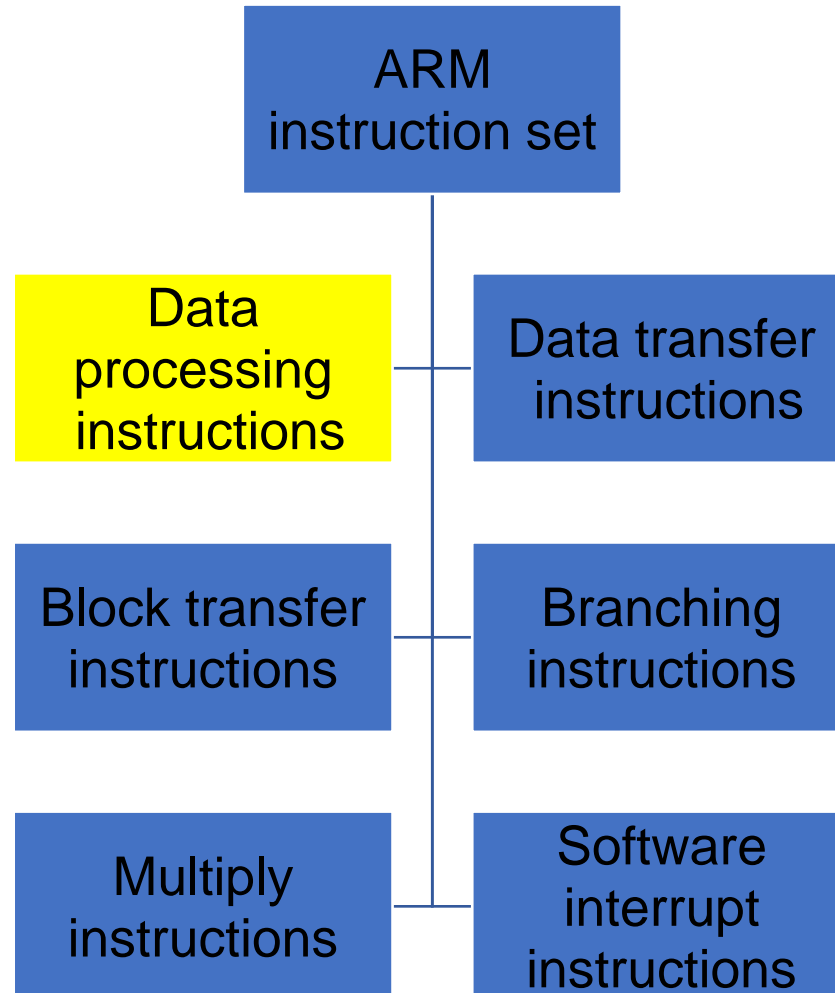
Data Movement Instruction

Arithmetic Instruction

Multiword Arithmetic



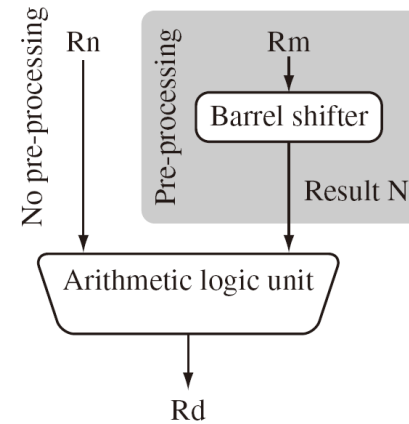
ARM INSTRUCTION SET



Microprocessor & Computer Architecture (μpCA)

Data processing instructions

- Largest family of ARM instructions, all sharing the same instruction format.
- Contains:
 - Arithmetic operations
 - Comparisons (no results - just set condition codes)
 - Logical operations
 - Data movement between registers
- Remember, this is a load / store architecture
 - These instructions only work on registers, **NOT** memory.
- They each perform a specific operation on one or two operands.
 - First operand always a register - Rn
 - Second operand sent to the ALU via barrel shifter.



- Operations are:
 - MOV operand2
 - MVN NOT operand2

Note that these make no use of operand1.

- Syntax:
 - <Operation>{<cond>}{S} Rd, Operand2
- Examples:
 - MOV r0, r1
 - MOVS r2, #10
 - MVNEQ r1,#0

Microprocessor & Computer Architecture (μpCA)

```
.text  
MOVS R0, #-256  
.end
```

```
R0      : -256  
R1      : 0  
R2      : 0  
R3      : 0  
R4      : 0  
R5      : 0  
R6      : 0  
R7      : 0  
R8      : 0  
R9      : 0  
R10 (s1) : 0  
R11 (fp) : 0  
R12 (ip) : 0  
R13 (sp) : 21504  
R14 (lr) : 0  
R15 (pc) : 70656
```

```
-----  
CPSR Register  
Negative (N) : 1  
Zero (Z)      : 0  
Carry (C)     : 0  
Overflow (V)  : 0
```

```
.text  
MOV R0,#5  
MOV R1,#5  
CMP R0,R1  
MVNEQ R2, #25  
.end
```

RegistersView	
General Purpose Floating Point	
Hexadecimal	
Unsigned Decimal	
Signed Decimal	
R0	: 0000000f
R1	: 0000000f
R2	: ffffffff0
R3	: 00000000
R4	: 00000000
R5	: 00000000
R6	: 00000000
R7	: 00000000
R8	: 00000000
R9	: 00000000
R10 (s1)	: 00000000
R11 (fp)	: 00000000
R12 (ip)	: 00000000
R13 (sp)	: 00005400
R14 (lr)	: 00000000
R15 (pc)	: 00011400

CPSR Register	
Negative (N)	: 0
Zero (Z)	: 1
Carry (C)	: 1
Overflow (V)	: 0

- Operations are:
 - ADD operand1 + operand2
 - SUB operand1 - operand2
 - RSB operand2 - operand1
- Syntax:
 - <Operation>{<cond>}{S} Rd, Rn, Operand2
- Examples
 - ADD r0, r1, r2
 - SUBGT r3, r3, #1
 - RSBLES r4, r5, #5

Microprocessor & Computer Architecture (μpCA)

Example: Data Processing Instructions

;ADD 2 numbers loaded from register
.text

MOV r0, #0x80

MOV r1, #20

ADD r2, r0, r1

;Sub 2 numbers loaded from register

SUB r4, r1, r0

SUB r3, r0, r1

.end

RegistersView	
General Purpose	Floating Point
Hexadecimal	
Unsigned Decimal	
Signed Decimal	
R0	: 128
R1	: 20
R2	: 148
R3	: 108
R4	: -108
R5	: 0
R6	: 0
R7	: 0
R8	: 0
R9	: 0
R10 (s1)	: 0
R11 (fp)	: 0
R12 (ip)	: 0
R13 (sp)	: 21504
R14 (lr)	: 0
R15 (pc)	: 70656

- **ADC:** $\text{Operand1} + \text{Operand 2} + \text{CPSR.c}$
- **SBC:** $\text{Operand 1} - \text{Operand2} - \text{NOT(CPSR.c)}$
or
 $\text{Operand1} - \text{Operand2} + \text{carry} - 1$
- **RSC:** $\text{Operand2} - \text{Operand 1} - \text{NOT(CPSR.c)}$
or
 $\text{Operand2} - \text{Operand1} + \text{carry} - 1$

NOTE: For SBC & RSC

Since Subtraction is performed using adder, the Operand2 & Carry in is inverted and fed to adder or Invert the Operand 2 first and subtract carry later.

- $\text{Operand1} + (-\text{Operand2})$
- $\text{Operand1} + \sim\text{Operand2} + 1$

Microprocessor & Computer Architecture (μpCA)

Arithmetic Operations With Carry



SBC: Operand1 - Operand2 + carry -1
 or
 Operand 1 – Operand2 –NOT(CPSR.c)

```
.text
MOV r0, #4
MOV r1, #2
SUB r3,r0,r1
SBC r2, r0, r1
.end
```

A screenshot of a 'RegistersView' window from a debugger. It has two tabs: 'General Purpose' and 'Floating Point'. Under 'General Purpose', there are three sub-tabs: 'Hexadecimal', 'Unsigned Decimal', and 'Signed Decimal'. The 'Signed Decimal' tab is selected and highlighted in blue. Below the tabs, a list of registers (R0 through R9) is shown with their corresponding values in decimal. R0 is 4, R1 is 2, R2 is 1, R3 is 2, and R4 through R9 are all 0.

RegistersView	
General Purpose Floating Point	
Hexadecimal	
Unsigned Decimal	
Signed Decimal	
R0	: 4
R1	: 2
R2	: 1
R3	: 2
R4	: 0
R5	: 0
R6	: 0
R7	: 0
R8	: 0
R9	: 0

~~**Wrong application of SBC**~~

Microprocessor & Computer Architecture (μpCA)

Arithmetic Operations With Carry

SBC: operand1 - operand2 + carry - 1

```
.text    MOV r0, #4
         MOV r1, #2
         MOV r4, #8
         MOV r5, #4
         SUBS r3,r0,r1
         SBC r2, r4, r5
.end
```

After executing SUBS

RegistersView	
General Purpose	Floating Point
Hexadecimal	
Unsigned Decimal	
Signed Decimal	
R0	: 4
R1	: 2
R2	: 0
R3	: 2
R4	: 8
R5	: 4
R6	: 0
R7	: 0
R8	: 0
R9	: 0
R10 (s1)	: 0
R11 (fp)	: 0
R12 (ip)	: 0
R13 (sp)	: 21504
R14 (lr)	: 0
R15 (pc)	: 4116

CPSR Register	
Negative (N)	: 0
Zero (Z)	: 0
Carry (C)	: 1
Overflow (V)	: 0

Microprocessor & Computer Architecture (μpCA)

Arithmetic Operations With Carry

```
.text    MOV r0, #4
         MOV r1, #2
         MOV r4, #8
         MOV r5, #4
         SUBS r3,r0,r1
         SBC r2, r4, r5
.end
```

RegistersView		🔍	✕
General Purpose		Floating Point	
Hexadecimal			
Unsigned Decimal			
Signed Decimal			
R0	: 4		
R1	: 2		
R2	: 4		
R3	: 2		
R4	: 8		
R5	: 4		
R6	: 0		
R7	: 0		

Microprocessor & Computer Architecture (μpCA)

Multiword Arithmetic



If more than 32 bit or 1 word operations need to be performed, **ADC**, **SBC**, **RSC** instructions are used.

- **Example 1:** 64 Bit Integer →

32 Bit MSB Word	32 Bit LSB Word
-----------------	-----------------
- Let **A** be a 64 bit data. **r0** will have least significant word and **r1** will have most significant word.
- Let **B** be a 64 bit data. **r2** will have least significant word and **r3** will have most significant word.
- Result of adding **A** and **B** will be in **r4** & **r5**.

ADDS r4,r0,r2 ; adding the least significant words

ADC r5,r1,r3 ; adding the most significant words

Example 2: These instructions subtract one 96-bit integer from another:

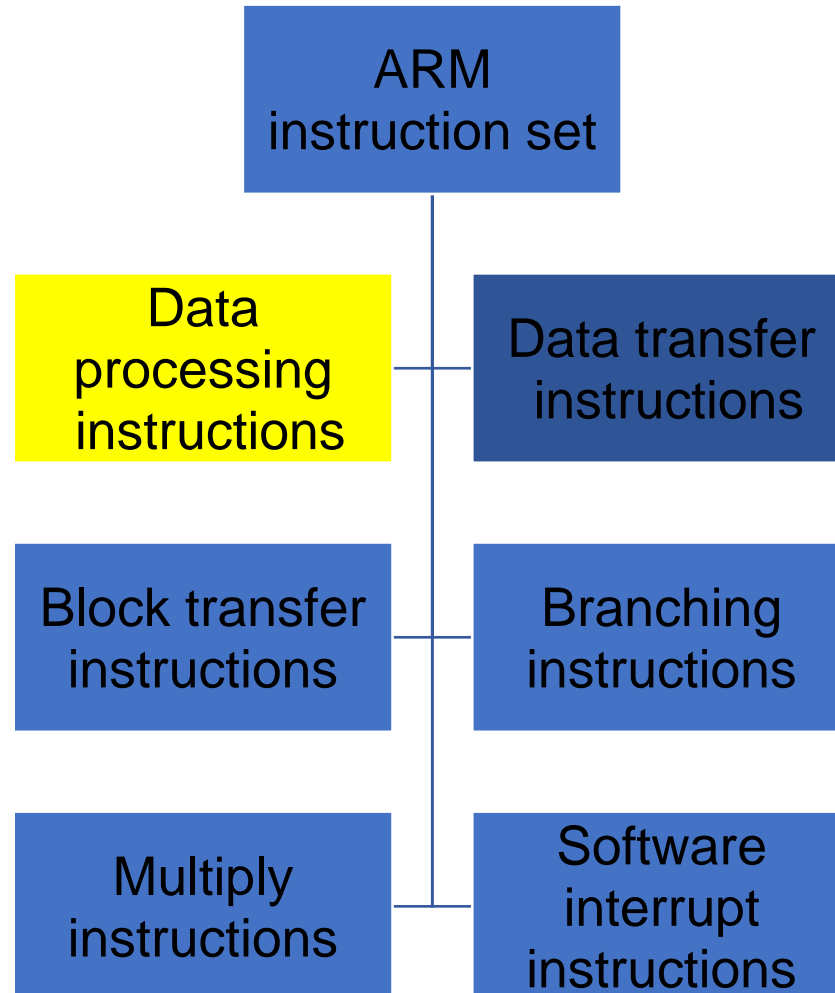
SUBS r3,r6,r9
SBCS r4,r7,r10
SBC r5,r8,r11

32 Bit MSB Word	32 Bit Word	32 Bit LSB Word
--------------------	----------------	--------------------

96 Bit Integer

Microprocessor & Computer Architecture (μpCA)

Next Class: Barrel Shifter





THANK YOU

Dr. D. C. Kiran

Department of Computer Science and Engineering

dckiran@pes.edu

9829935135