Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

PES UNIVERSITY ONLINE

Microprocessor & Computer Architecture (µpCA)

Data Processing Instructions

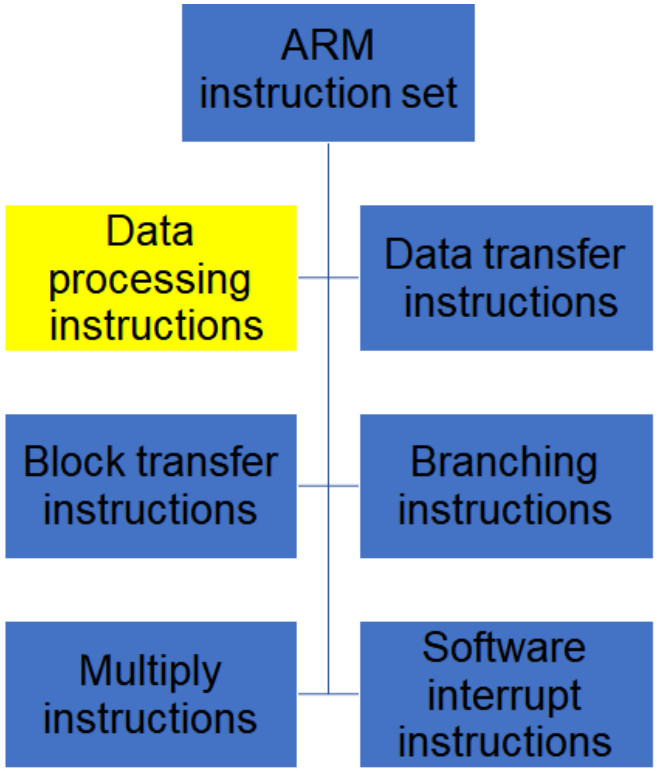
Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

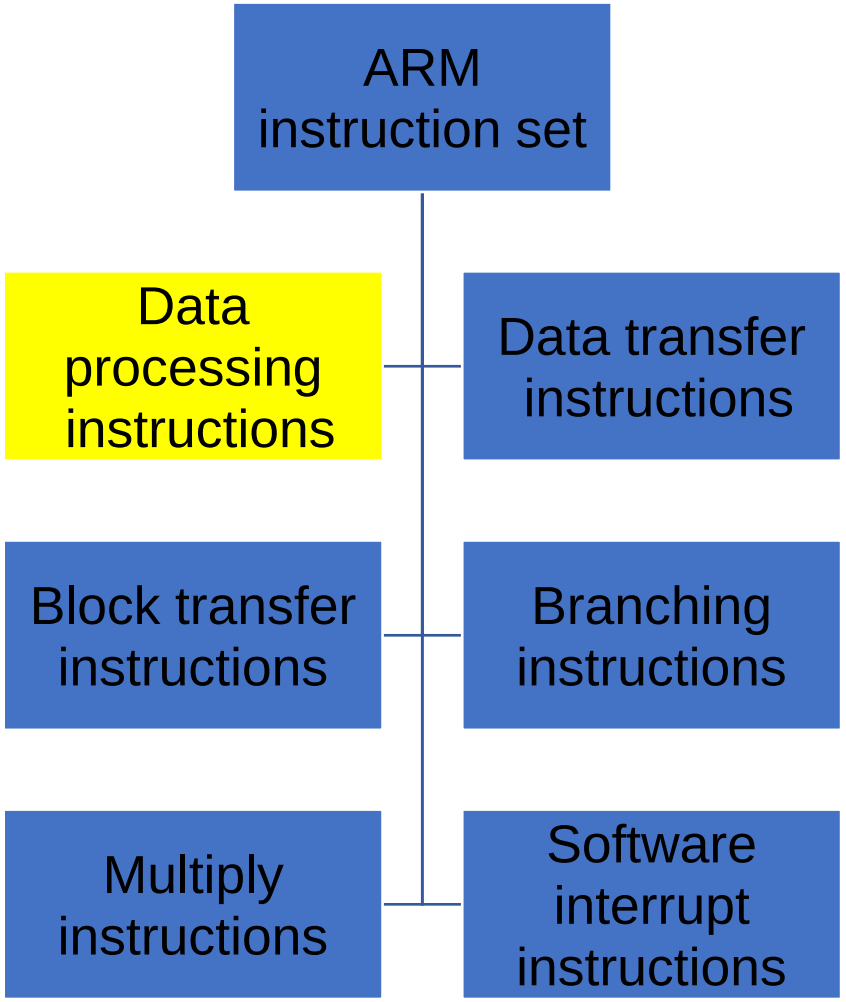
Syllabus

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

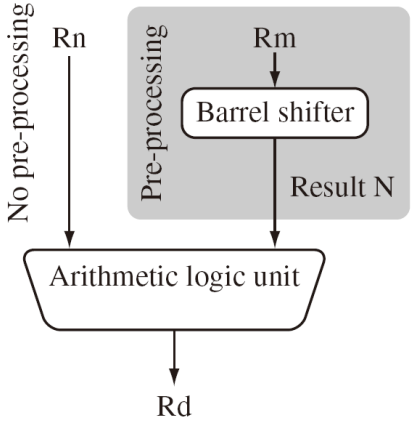


ARM INSTRUCTION SET



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 instruction 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.

Data Movement

- Operations are:
  - MOV operand2
  - MVN NOT operand2

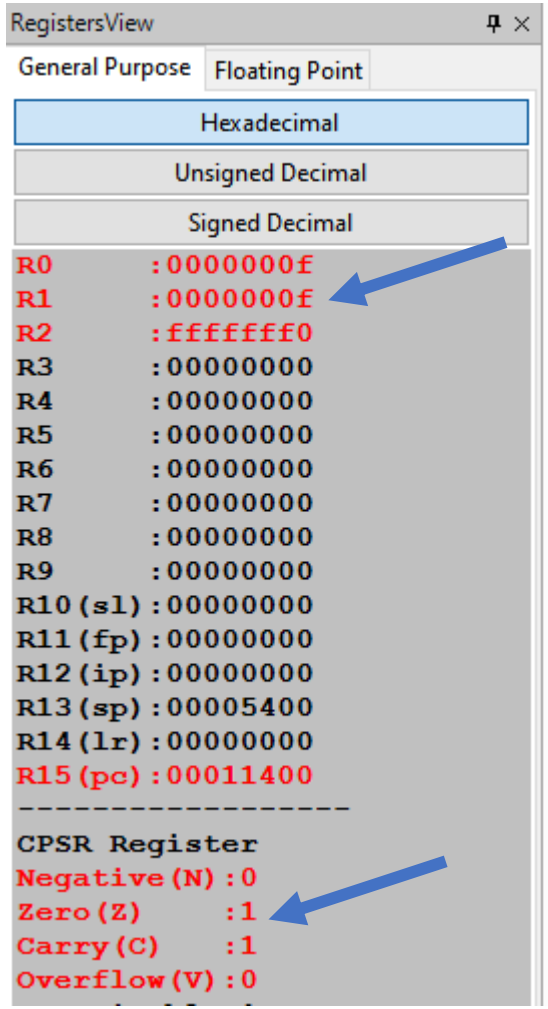
  Note that these make no use of operand1.
- Syntax:
  - <Operation>{<cond>}{S} Rd, Operand2
- Examples:
  - MOV r0, r1
  - MOVS r2, #10
  - MVNEQ r1,#0

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(sl):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
```

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



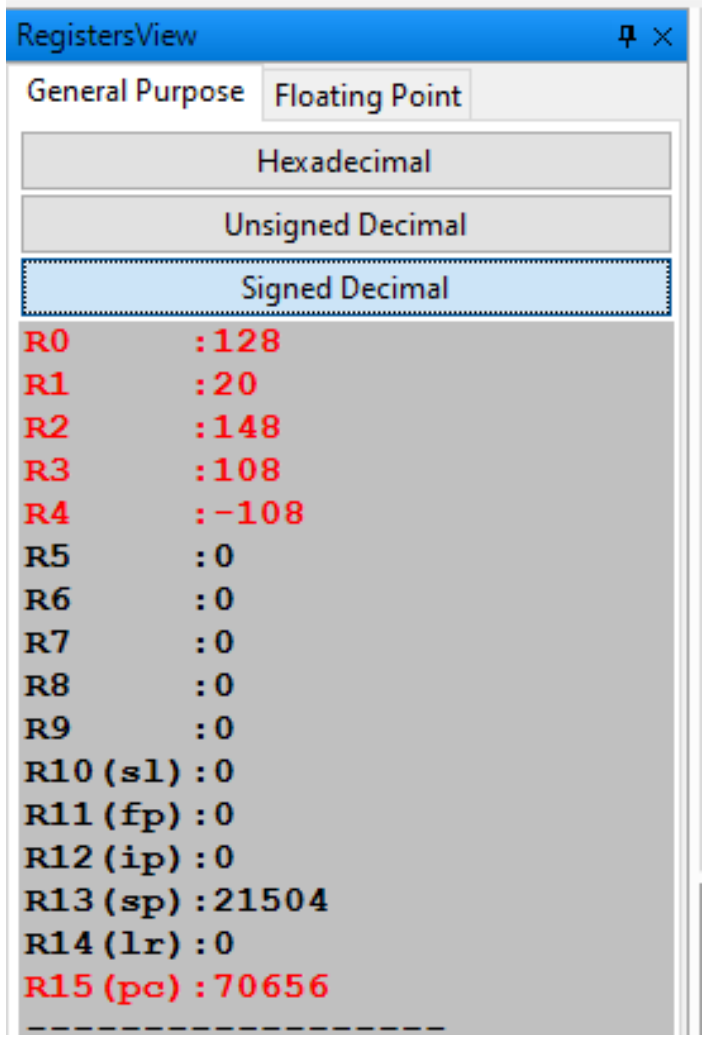
Arithmetic Operations

- 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

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
```



Microprocessor & Computer Architecture (µpCA)

Arithmetic Operations With Carry

- **ADC:** Operand1 + Operand 2 + CPSR.c

- **SBC:** Operand 1 – Operand2 –NOT(CPSR.c)

  or

  Operand1 - Operand2 + carry -1

- **RSC:** Operand2-Operand 1 - NOT(CPSR.c)

  or

  Operand2 – Operand1 + 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.

- Operand1+(-Operand2)
- Operand1+~Operand2+1

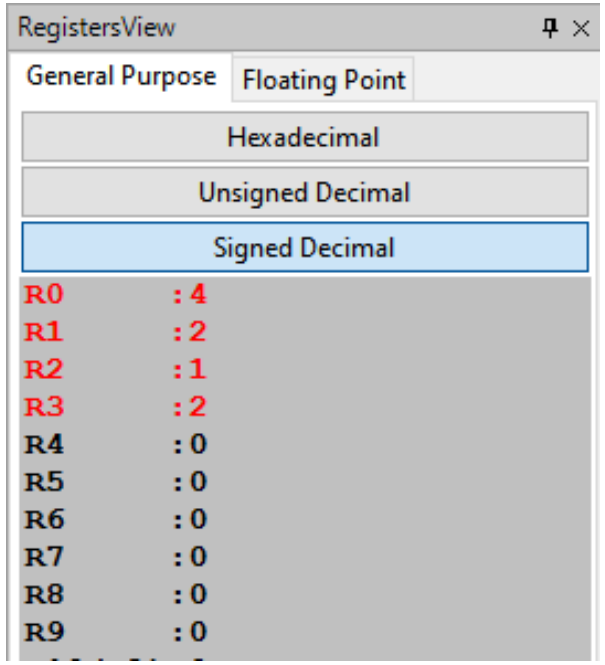
Arithmetic Operations With Carry

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
```



~~Wrong application of SBC~~

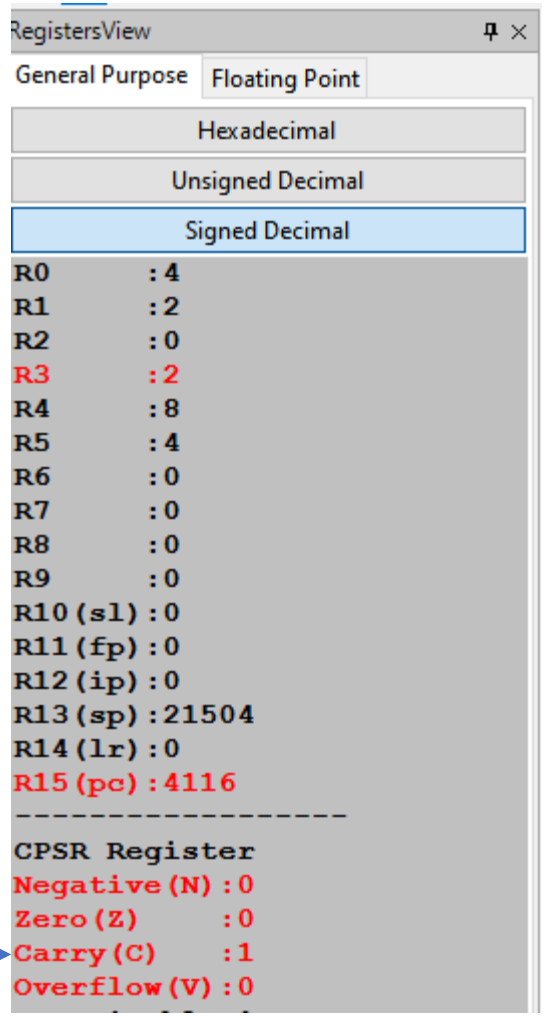
Microprocessor & Computer Architecture (µpCA)

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
```

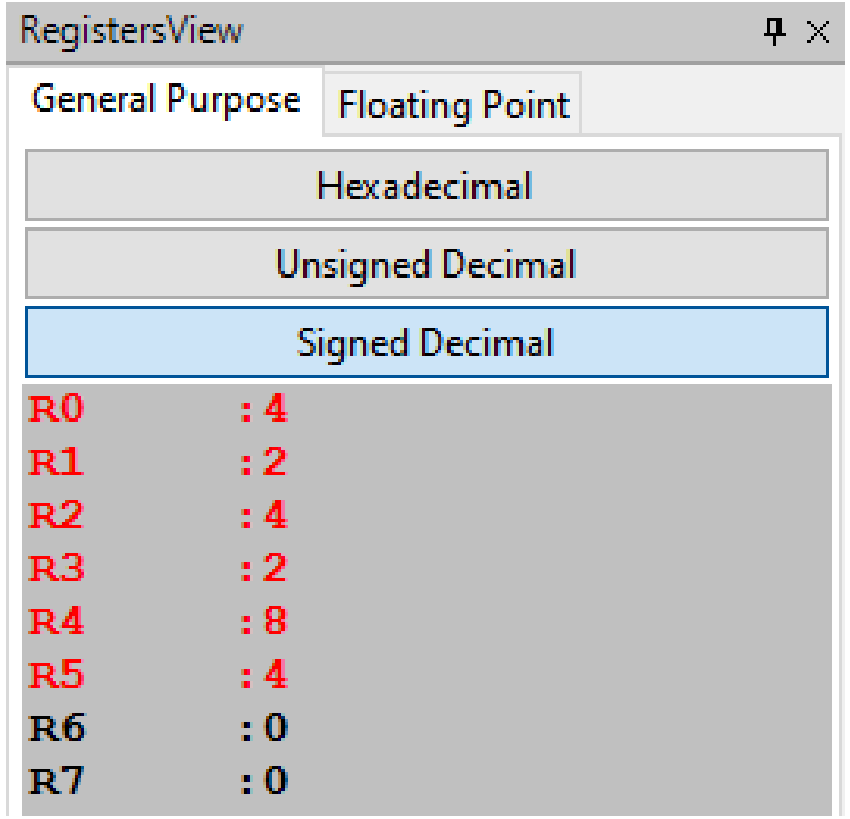
After executing SUBS



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
```



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:

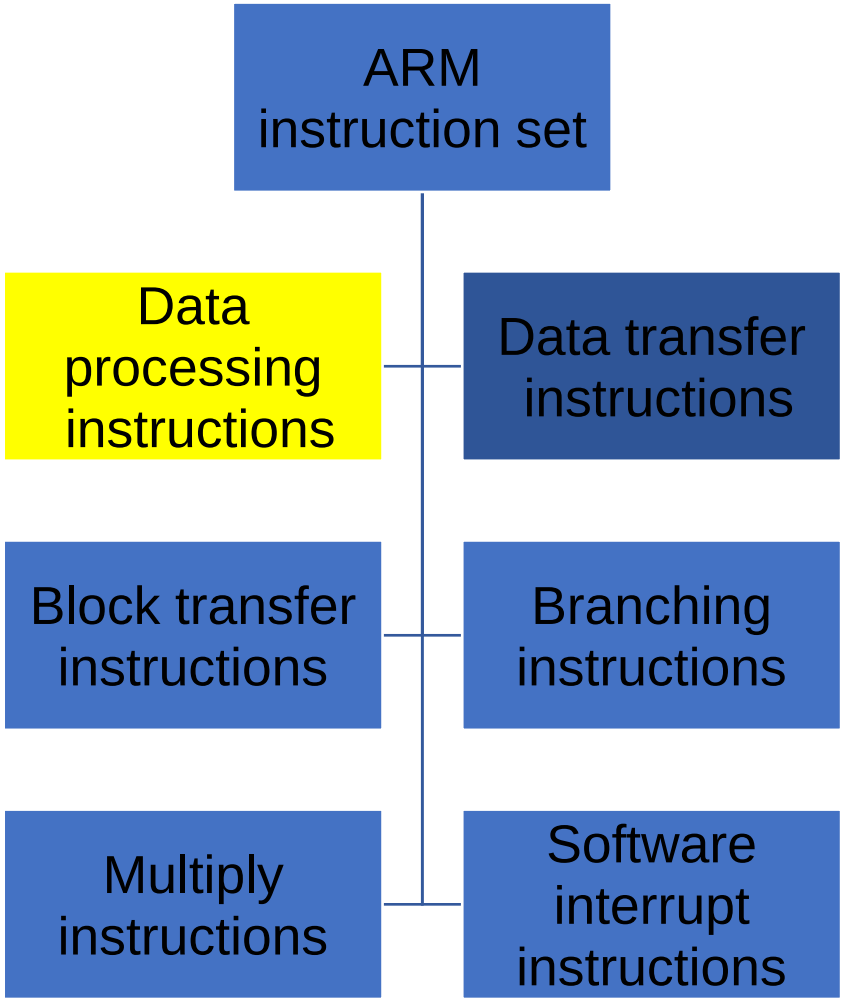
```
SUBS r3,r6,r9
SBCS r4,r7,r10
SBC r5,r8,r11
```

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