Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

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

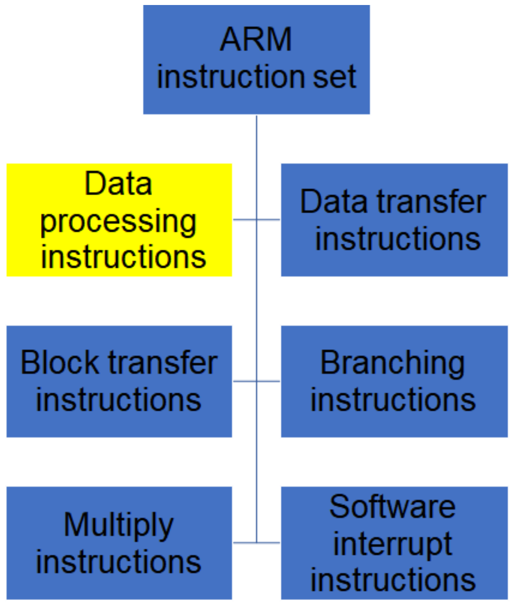
  ~~Data Movement Instruction~~

  ~~Arithmetic Instruction~~

  ~~Multiword Arithmetic~~

  ~~Barrel Shifter~~

  Logical and Comparison Instruction


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

Comparison Instruction

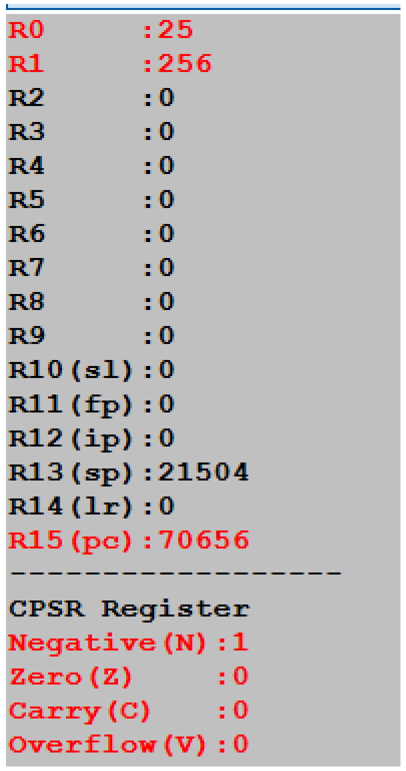
Syntax:

<Operation>{<cond>} Rn, Operand2

```
• CMP   R1, R2        @ set cc on R1-R2

• CMN   R1, R2        @ set cc on R1+R2

• TST   R1, R2        @ set cc on R1 and R2

• TEQ   R1, R2        @ set cc on R1 xor R2
```

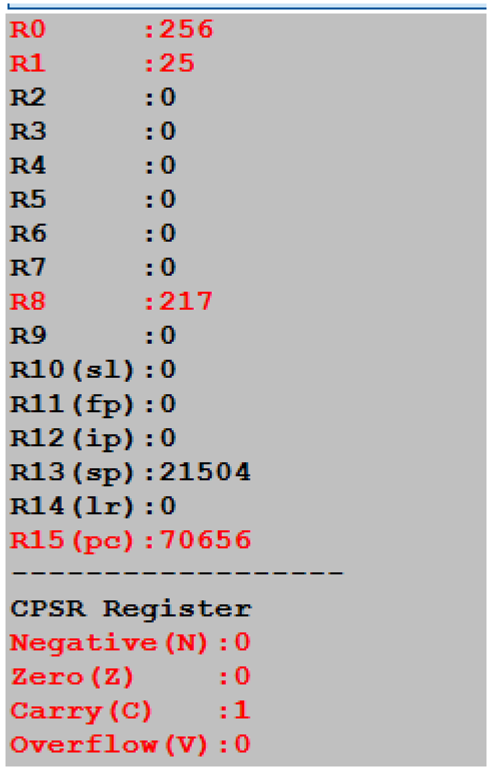
Microprocessor & Computer Architecture (μpCA)

```
.text
        MOV R0, #25
        MOV R1, #256
        CMP R0,R1
.end
```



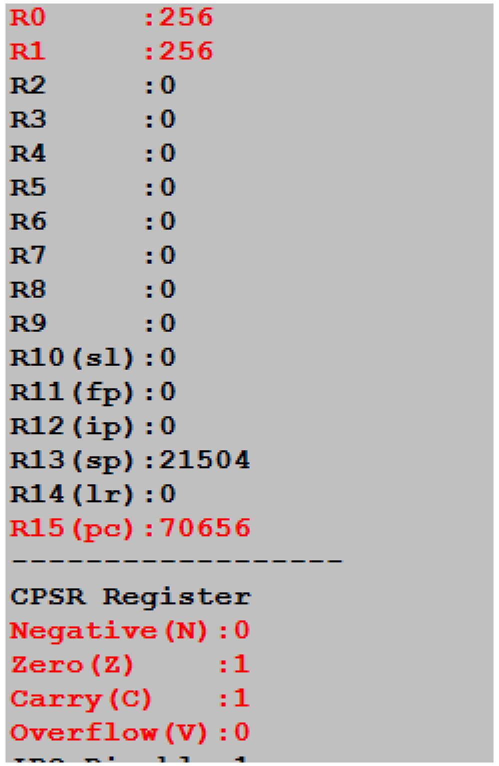
```
R0       :25
R1       :256
R2       :0
R3       :0
R4       :0
R5       :0
R6       :0
R7       :0
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
-------------------
CPSR Register
Negative(N):1
Zero(Z)     :0
Carry(C)    :0
Overflow(V):0
```

```
.text
        MOV R0, #256
        MOV R1, #25
        CMP R0,R1
.end
```



```
R0       :256
R1       :25
R2       :0
R3       :0
R4       :0
R5       :0
R6       :0
R7       :0
R8       :217
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
-------------------
CPSR Register
Negative(N):0
Zero(Z)     :0
Carry(C)    :1
Overflow(V):0
```

```
.text
        MOV R0, #256
        MOV R1, #256
        CMP R0,R1
.end
```

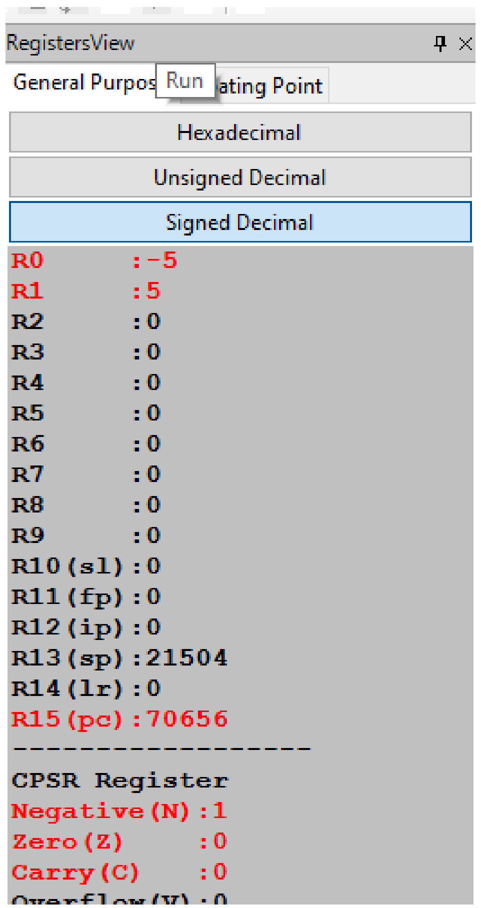


```
R0       :256
R1       :256
R2       :0
R3       :0
R4       :0
R5       :0
R6       :0
R7       :0
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
-------------------
CPSR Register
Negative(N):0
Zero(Z)     :1
Carry(C)    :1
Overflow(V):0
```

Microprocessor & Computer Architecture (μpCA)

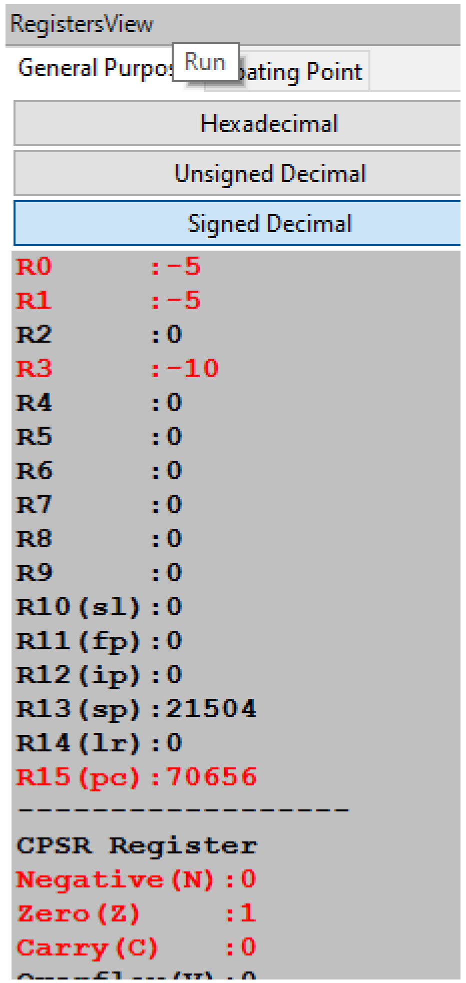
Test (TST) & Test Equivalence (TEQ)

```
.text
MOV R0,#-5
MOV R1,#5
TEQ R0,R1
ADDEQ R3,R0,R1
.end
```



```
R0       :-5
R1       :5
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
Overflow(V):0
```

```
.text
MOV R0,#-5
MOV R1,#-5
TEQ R0,R1
ADDEQ R3,R0,R1
.end
```



```
R0       :-5
R1       :-5
R2       :0
R3       :-10
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):0
Zero(Z)    :1
Carry(C)   :0
Overflow(V):0
```

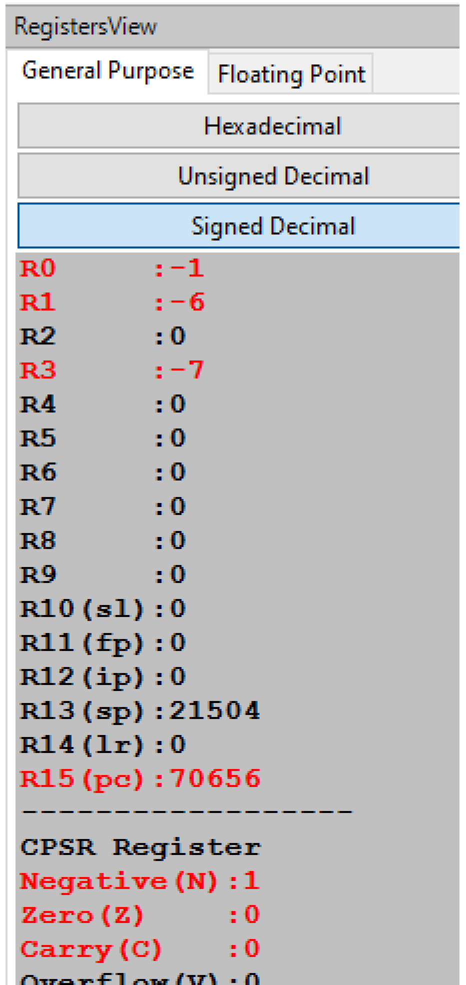
Test (TST) & Test Equivalence (TEQ)

TST perform AND operation against Operand 1 and Operand 2
Update the flags of CPSR based on the register
Used to check if any flag is set.

Example1 :

```
.text
MOV R0,#-1
MOV R1,#-6
TST R0,R1
ADDMI R3,R0,R1
.end
```

$$\begin{array}{r} 1111 \\ \underline{1010} \\ \underline{1010} \end{array}$$

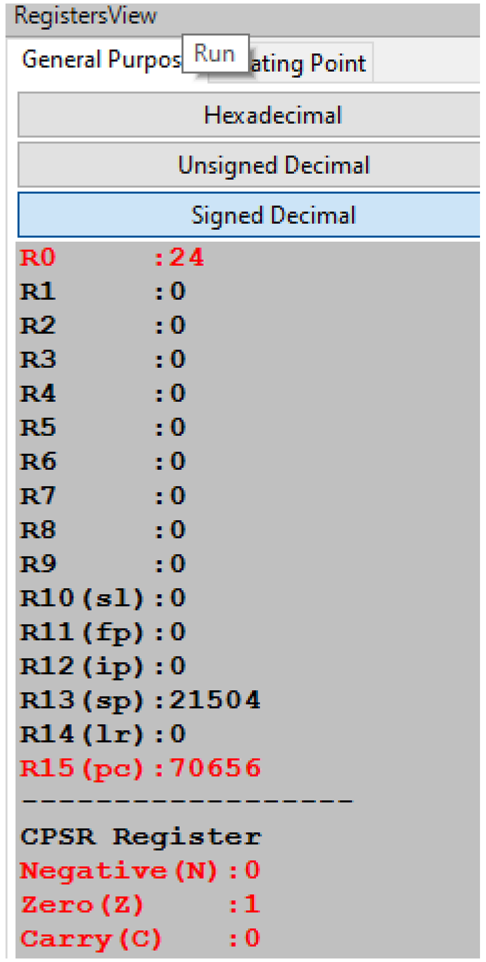


```
R0       :-1
R1       :-6
R2       :0
R3       :-7
R4       :0
R5       :0
R6       :0
R7       :0
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
-------------------
CPSR Register
Negative(N):1
Zero(Z)     :0
Carry(C)    :0
Overflow(V):0
```

Test (TST) & Test Equivalence (TEQ)

What do you infer from the following program?

```
.text
MOV R0,#24
TST R0,#1
.end
```



```
R0       :24
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):0
Zero(Z)    :1
Carry(C)   :0
```

Logical Operations

- Operations are:
  - AND
  - EOR
  - ORR
  - BIC
- Syntax:
  - <Operation>{<cond>}{S} Rd, Rn, Operand2
- Examples:
  - AND r0, r1, r2
  - BICEQ r2, r3, #7
  - EORS r1,r3,r0

Microprocessor & Computer Architecture (μpCA)

Logical Operations

```
• AND  R0, R1, R2    @ R0 = R1 and R2
• ORR  R0, R1, R2    @ R0 = R1 or  R2
• EOR  R0, R1, R2    @ R0 = R1 xor R2
• BIC  R0, R1, R2    @ R0 = R1 and (~R2)
```

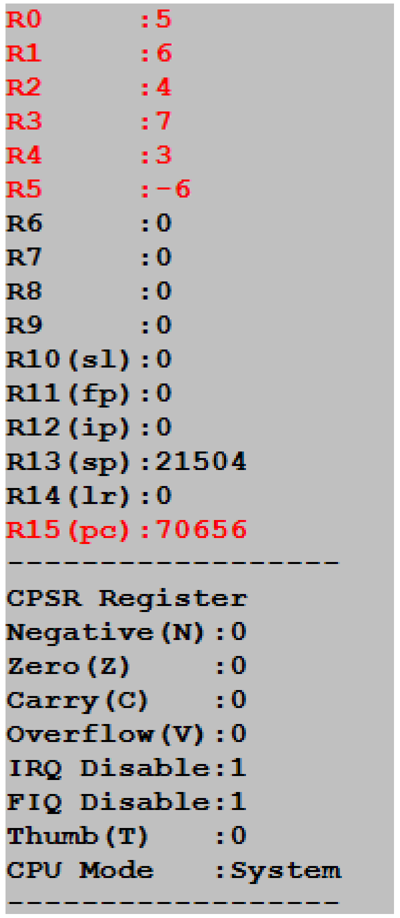
bit clear: **R2** is a mask identifying which bits of **R1** will be cleared to zero

```
R1=0x11111111     R2=0x01100101

BIC R0, R1, R2

R0=0x10011010
```

Microprocessor & Computer Architecture (μpCA)

Example: Logical Operation

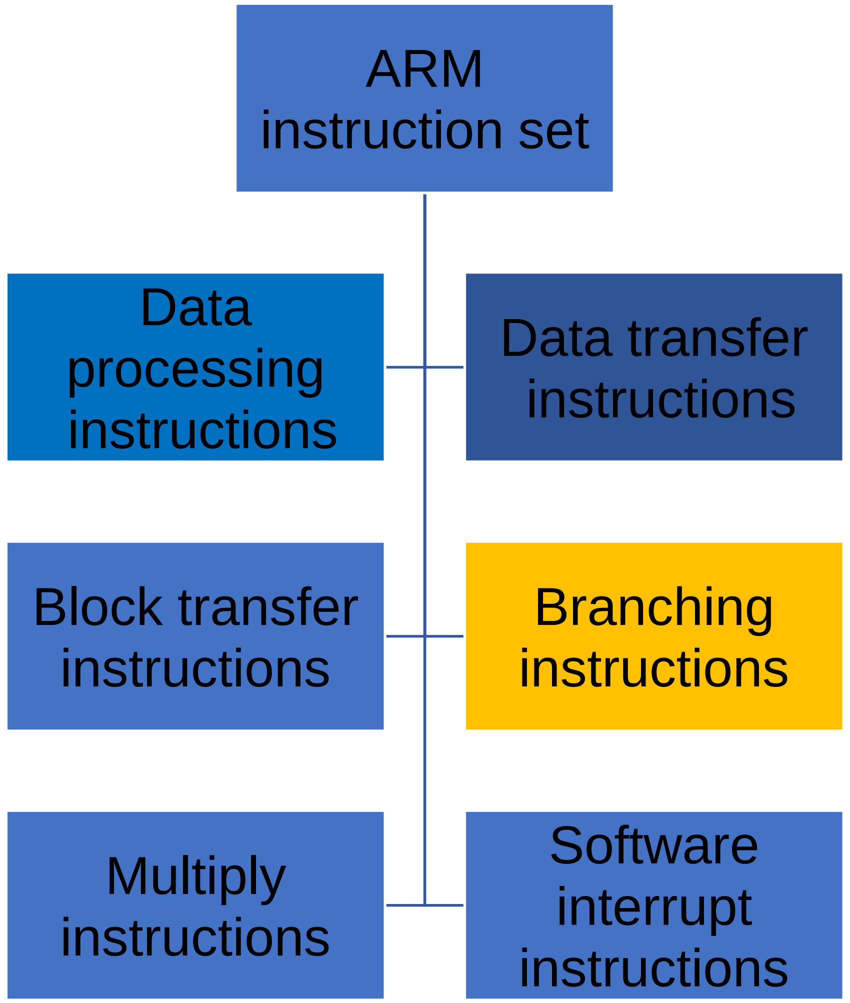
MOV R0,#5
MOV R1,#6
AND R2,R0,R1 ; Logical AND
ORR R3,R0,R1 ; Logical OR
EOR R4,R0,R1 ; Logical XOR
MVN R5,R0 ; Complemented value is moved to R5



```
R0      :5
R1      :6
R2      :4
R3      :7
R4      :3
R5      :-6
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):0
Zero(Z)    :0
Carry(C)   :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
```

Microprocessor & Computer Architecture (μpCA)

Next Class


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

THANK YOU

Dr. D. C. Kiran

Department of Computer Science and Engineering

dckiran@pes.edu

9829935135