Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Multiplication, PSR & SWAP 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~~
  - ~~Flow Control Instructions~~
  - ~~Data Transfer Instructions~~
  - ~~Block Transfer Instructions (Stack & Procedure Call)~~
  - Multiplication
  - MSR & MRS Instructions
  - Swap

Multiplication

MUL	**Multiply**	**32-bit result**
MLA	**Multiply accumulate**	**32-bit result**
UMULL	**Unsigned multiply**	**64-bit result**
UMLAL	**Unsigned multiply accumulate**	**64-bit result**
SMULL	**Signed multiply**	**64-bit result**
SMLAL	**Signed multiply accumulate**	**64-bit result**

Multiplication

MUL{<cond>}{S}Rd,Rf,Rs ;Rd=(RfxRs)$_{[31:0]}$

- **MUL R0, R1, R2 @ R0 = (R1xR2)$_{[31:0]}$**

- Features:
  - **Second operand** can't be immediate
  - The result register must be different from the first operand

    R**d**≠R**f**

Multiplication

Hexadecimal
Unsigned Decimal
Signed Decimal

R0	:25
R1	:5
R2	:125
R3	:0
R4	:0
R5	:0
R6	:0
R7	:0
R8	:0
R9	:0
R10(sl)	:0
R11(fp)	:0
R12(ip)	:0

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

Multiplication c=c+a[i]*b[i]

```
                        .TEXT
00001000:E59F5030       LDR R5,=A
00001004:E59F6030       LDR R6,=B
00001008:E59F7030       LDR R7,=C
0000100C:E3A03003       Mov R3,#3     ;count Of Numbers
00001010:E3A04000       MOV R4,#0
00001014:               Loop:
00001014:E4951004               Ldr R1,[R5],#4
00001018:E4962004               Ldr R2,[R6],#4
0000101C:E0000291               Mul R0,R1,R2     ;multiple R1 And R2 And Store Result In R0
00001020:E0844000               Add R4,R4,R0
00001024:E4874004               STR R4,[R7],#4
00001028:E2433001               Sub R3,R3,#1
0000102C:E3330000               Teq R3,#0        ; Test For Equality
00001030:1AFFFFF7               Bne Loop
00001034:EF000011               Swi 0x11       ; Interrupt For Termination Of Program

                        .DATA
00001044:               A: .word 1,3,5
00001050:               B: .word 2,4,6
0000105C:               C: .word
```

Multiplication c=c+a[i]*b[i]

R0	:00000000
R1	:00000000
R2	:00000000
R3	:00000005
R4	:00000000
R5	:00001044
R6	:00001050
R7	:0000105c
R8	:00000000

R4=0

R0	:00000002
R1	:00000001
R2	:00000002
R3	:00000005
R4	:00000002
R5	:00001048
R6	:00001054
R7	:0000105c
R8	:00000000

=2

R4=0+2

R0	:0000000c
R1	:00000003
R2	:00000004
R3	:00000004
R4	:0000000e
R5	:0000104c
R6	:00001058
R7	:00001060
R8	:00000000

=14

R4=2+12

R0	:0000001e
R1	:00000005
R2	:00000006
R3	:00000003
R4	:0000002c
R5	:00001050
R6	:0000105c
R7	:00001064
R8	:00000000

=44

R4=14+30

MemoryView1

0000105C

0000105c	0002	0000	000E	0000	002C	0000
00001080	8181	8181	8181	8181	8181	8181

Multiply-Accumulate c=c+a[i]*b[i]

<MLA> Rd, Rf, Rn, Rm @ Rd= RfxRn+Rm

- **MLA R4, R1, R2, R4 @ R4 = R1xR2+R4**

```
                        .TEXT
00001000:E3A05D41       LDR R5,=A
00001004:E59F6028       LDR R6,=B
00001008:E59F7028       LDR R7,=C
0000100C:E3A03003       Mov R3,#3     ;count Of Numbers
00001010:E3A04000       MOV R4,#0
00001014:               Loop:
00001014:E4951004               Ldr R1,[R5],#4
00001018:E4962004               Ldr R2,[R6],#4
0000101C:E0244291               MlA R4,R1,R2,R4     ;multiple R1 And R2 And Store Result In R4

00001020:E4874004               STR R4,[R7],#4
00001024:E2433001               Sub R3,R3,#1
00001028:E3330000               Teq R3,#0        ; Test For Equality
0000102C:1AFFFFF8               Bne Loop
00001030:EF000011               Swi 0x11       ; Interrupt For Termination Of Program

                        .DATA
00001040:               A: .word 1,3,5
0000104C:               B: .word 2,4,6
00001058:               C: .word
```

Multiply & Multiply with Accumulate to Produce 64-bit Result

UMULL	**Unsigned multiply**	**64-bit result**
UMLAL	**Unsigned multiply accumulate**	**64-bit result**
SMULL	**Signed multiply**	**64-bit result**
SMLAL	**Signed multiply accumulate**	**64-bit result**

UMULL R0, R4, R5, R6 ; Multiplies R5 and R6, writes the top 32 bits to R4 ; and the bottom 32 bits to R0

UMLAL R3, R6, R2, R7 ; Multiplies R2 and R7, adds R6, adds R3, writes the ; top 32 bits to R6, and the bottom 32 bits to R3

Multiply & Multiply with Accumulate to Produce 64-bit Result

Syntax: <SMLAL/SMULL/UMLAL/UMULL>{cond}{S}**RdLo**, RdHi, Rm,Rs

SMLAL	Signed multiply accumulate Long	[Rdhi, RdLo]=[RdHi,RdLo]+(Rm*Rs)
SMULL	Signed multiply Long	[Rdhi, RdLo]= (Rm*Rs)
UMLAL	Unsigned Multiply accumulate Long	[Rdhi, RdLo]=[RdHi,RdLo]+(Rm*Rs)
UMULL	Unsigned Multiply Long	[Rdhi, RdLo]= (Rm*Rs)

Microprocessor & Computer Architecture (µpCA)

Multiply & Multiply with Accumulate to Produce 64-bit Result

```
.text
ldr r0,=a
ldr r1,=b
ldr r2,[r0]
ldr r3,[r1]

umull r4,r5,r2,r3
.data
a:.word 20000000
b:.word 30000000
```

RegistersView

General Purpose | Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

```
R0        :4124
R1        :4128
R2        :20000000
R3        :30000000
R4        :1658683392
R5        :139698
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
-----------------
```

smul.s

```
                          .text
00001000:E59F000C         ldr r0,=a
00001004:E59F100C         ldr r1,=b
00001008:E5902000         ldr r2,[r0]
0000100C:E5913000         ldr r3,[r1]

00001010:E0854392         umull r4,r5,r2,r3
                          .data
0000101C:                 a:.word 20000000
00001020:                 b:.word 30000000
```

[Rdhi, RdLo]= (Rm*Rs)

R4=Rdhi
R5= RdLo

MemoryView0

PSR Instructions

Move to Register from Status Register (MRS): Read from either CPSR or SPSR

Example:

MRS R0, CPSR

MRS R1, SPSR

Move to Status Register from Register (MSR):

Example:

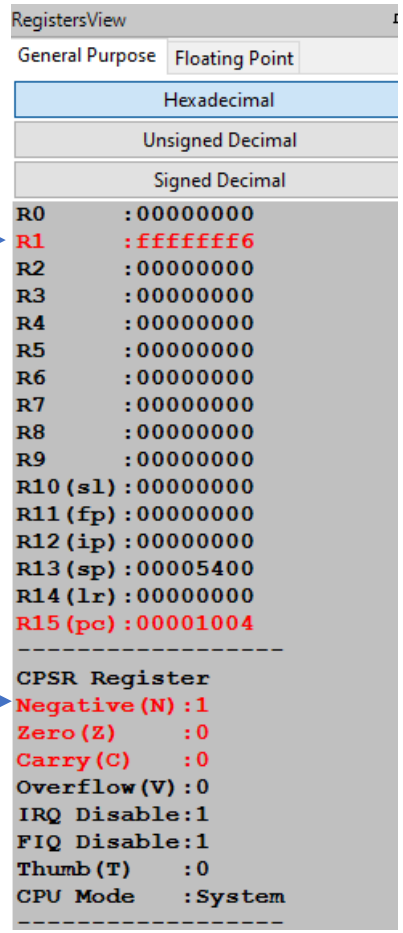
```
MSR CPSR_field, R0
MSR SPSR_field, R1
```

```
_field

_c: Control Field (0:7)
_f: Flag Field(24:31)
_x: Extension (8:15)
_s:Status (16:23)
```

Reference

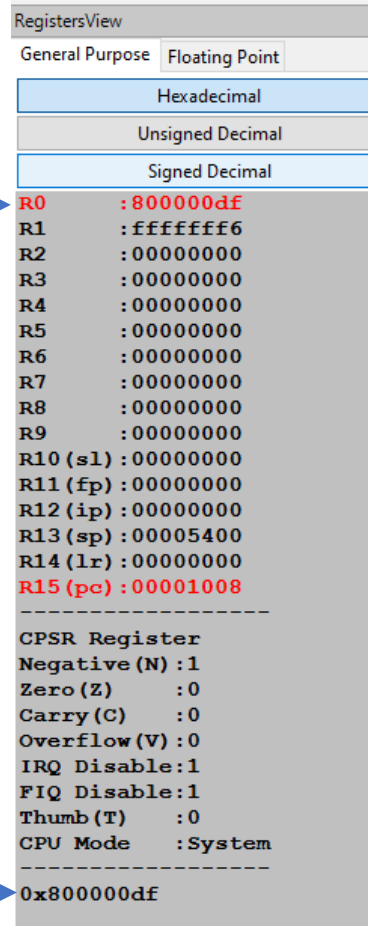
PSR Instructions

```
.text
MOVS R1,#-10
MRS R0,CPSR          ; Take a copy of the CPSR.
AND R0,R0,#0000      ; Clear the mode bits.
MSR CPSR_F,R0        ; Write back the modified CPSR.
.end
```



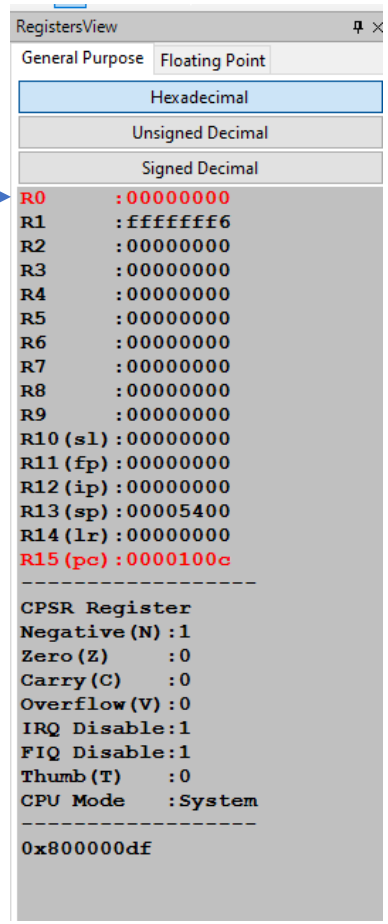
PSR Instructions

```
.text
MOVS R1,#-10
MRS R0,CPSR          ; Take a copy of the CPSR.
AND R0,R0,#0000      ; Clear the mode bits.
MSR CPSR_F,R0        ; Write back the modified CPSR.
.end
```



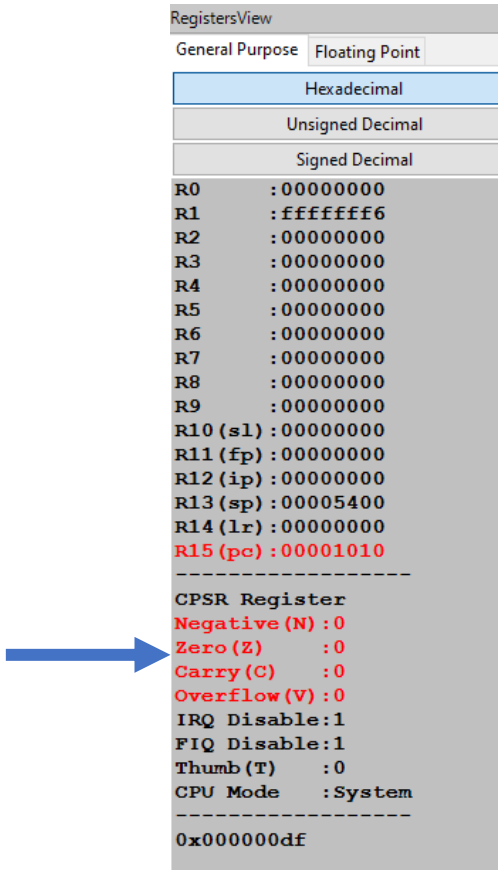
PSR Instructions

```
.text
MOVS R1,#-10
MRS R0,CPSR          ; Take a copy of the CPSR.
AND R0,R0,#0000      ; Clear the mode bits.
MSR CPSR_F,R0        ; Write back the modified CPSR.
.end
```

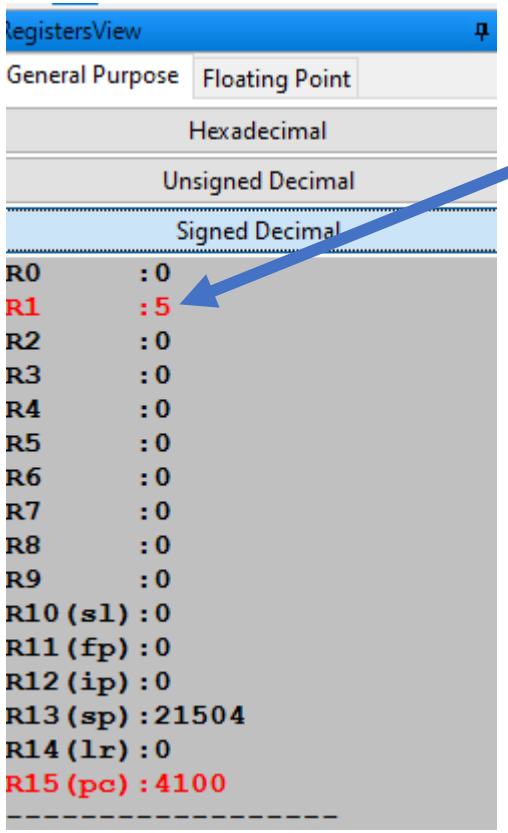


PSR Instructions

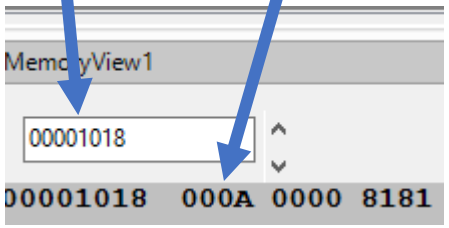
```
.text
MOVS R1,#-10
MRS R0,CPSR          ; Take a copy of the CPSR.
AND R0,R0,#0000      ; Clear the mode bits.
MSR CPSR_F,R0        ; Write back the modified CPSR.
.end
```



SWAP :A & R1




```
                        .text

00001000:E3A01005       MOV R1,#5
00001004:E59F2008       LDR R2,=a
00001008:E5923000       LDR R3,[R2]
0000100C:E5821000       STR R1,[R2]
00001010:E1A01003       MOV R1,R3
                        .data

00001018:               a:.word 10
```



SWAP :A & R1

RegistersView

General Purpose | Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

```
R0        :0
R1        :5
R2        :4120
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):4112
-----------------
```

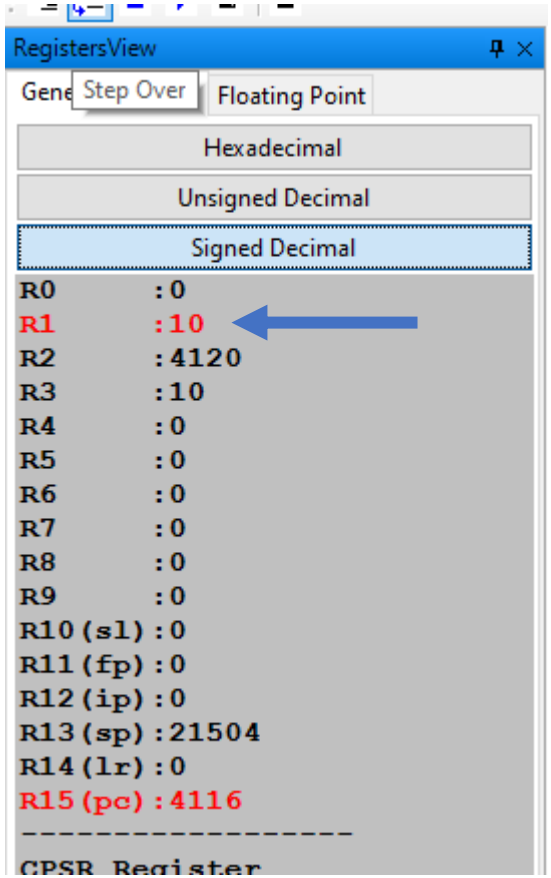
```
                        .text

00001000:E3A01005       MOV R1,#5
00001004:E59F2008       LDR R2,=a
00001008:E5923000       LDR R3,[R2]
0000100C:E5821000       STR R1,[R2]
00001010:E1A01003       MOV R1,R3
                        .data

00001018:               a:.word 10
```

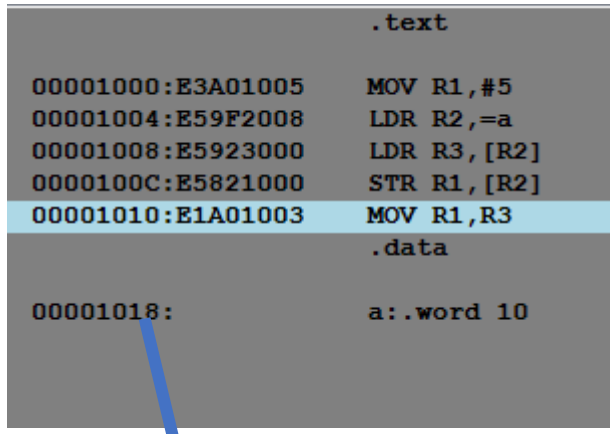
MemoryView1

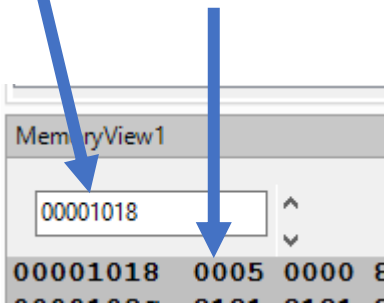
00001018

```
00001018  0005 0000 81
0000103C  8181 8181 81
```

SWAP :A & R1


RegistersView
Gene
Step Over
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0
R1 :10
R2 :4120
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):4116
CPSR Register


.text
00001000:E3A01005 MOV R1,#5
00001004:E59F2008 LDR R2,=a
00001008:E5923000 LDR R3,[R2]
0000100C:E5821000 STR R1,[R2]
00001010:E1A01003 MOV R1,R3
.data
00001018: a:.word 10


MemoryView1
00001018
00001018 0005 0000

SWAP instruction

SWP <Swap Destination>, <Original>, [<address>]

```
.text
LDR R2,=a
MOV R1,#5
SWP R1,R1,[R2]
.data
a:.word 10
```

RegistersView

General Purpose | Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

```
R0        :0
R1        :10
R2        :4112
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
```

```
                        .text
00001000:E59F2004       LDR R2,=a
00001004:E3A01005       MOV R1,#5
00001008:E1021091       SWP R1,R1,[R2]
                        .data

00001010:               a:.word 10
```

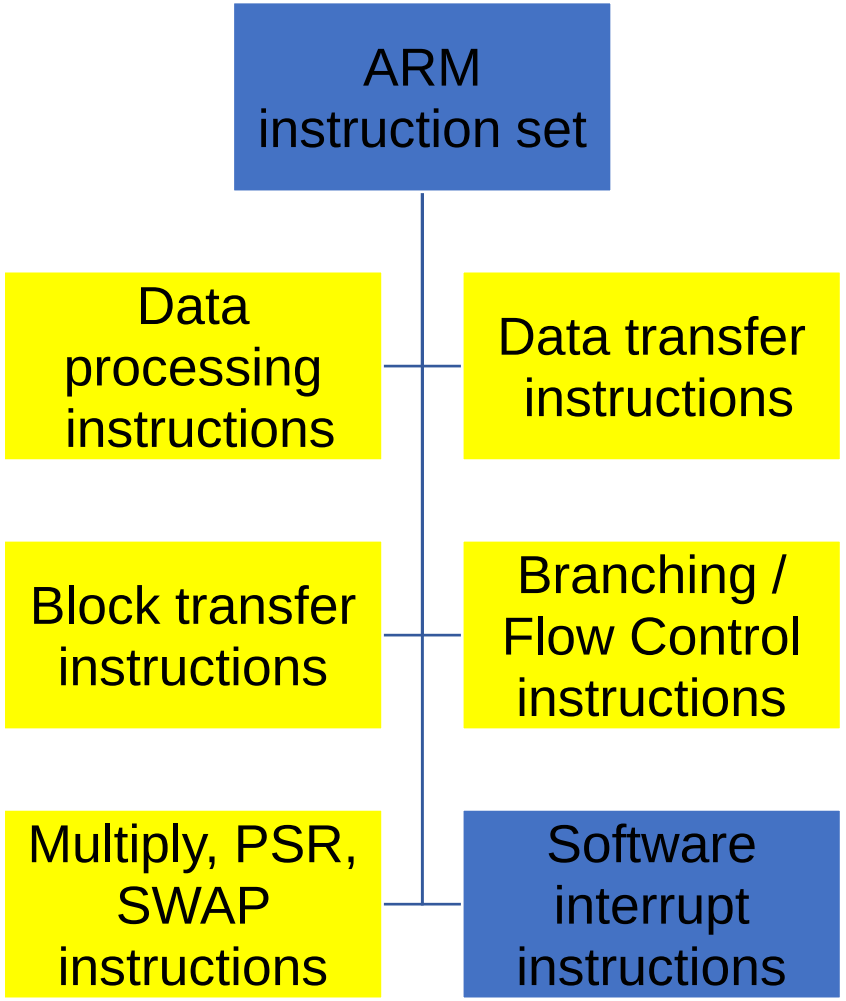
After Swapping

MemoryView1

00001010

```
00001010  0005 0000 8181 8
```

Next Session



THANK YOU

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

dckiran@pes.edu

9829935135