Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Multiple Register Load / Store
or
Block Transfer Instructions

Dr. D. C. Kiran

Department of Computer Science and Engineering

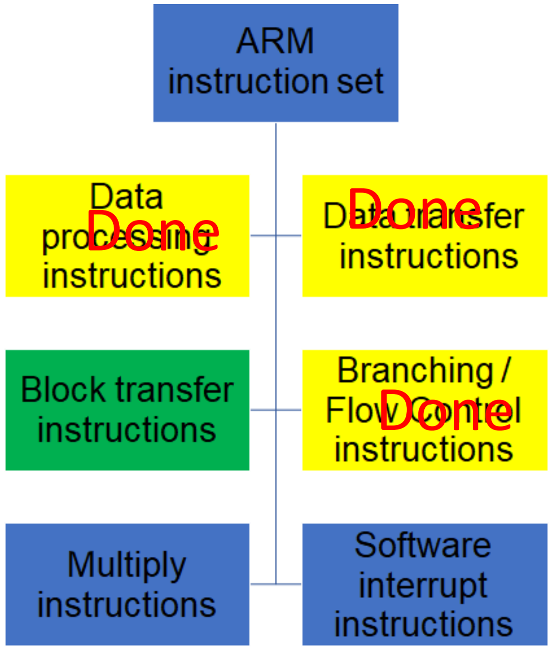
Microprocessor & Computer Architecture (μpCA)

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~~

Multiple Register Load / Store (Block Transfer Instructions)



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

Single Register Load and Store

Consider a Scenario where you need to transfer, 10,20,30 to register R1,R2,R3

```
.text
LDR R4,=A
LDR R1,[R4],#4
LDR R2,[R4],#4
LDR R3,[R4],#4
.data
A:.word 10,20,30,40,50
```

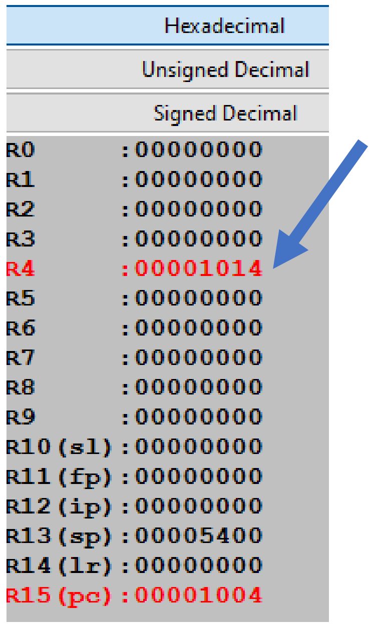
Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

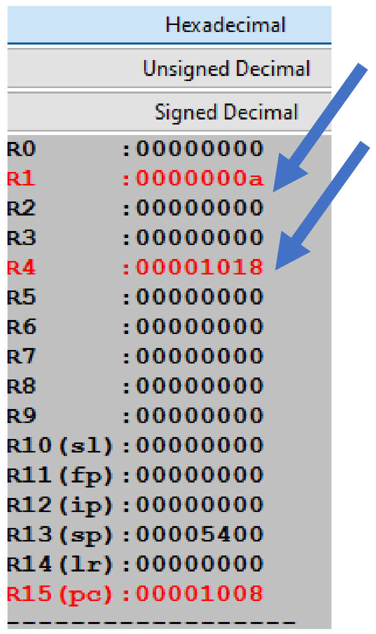
Single Register Load and Store

```
         .text
          LDR R4,=A
          LDR R1,[R4],#4
          LDR R2,[R4],#4
          LDR R3,[R4],#4
         .data
00001014  A:.word 10,20,30,40,50
```

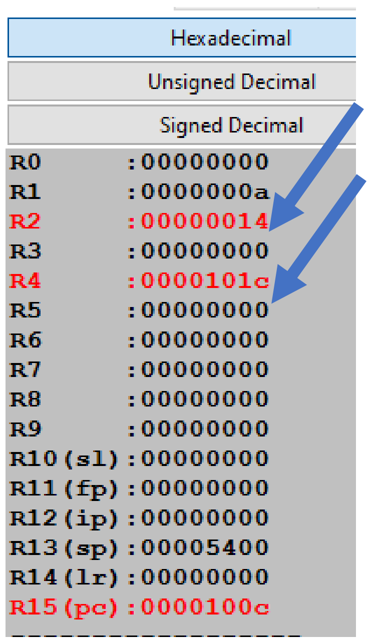
Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60



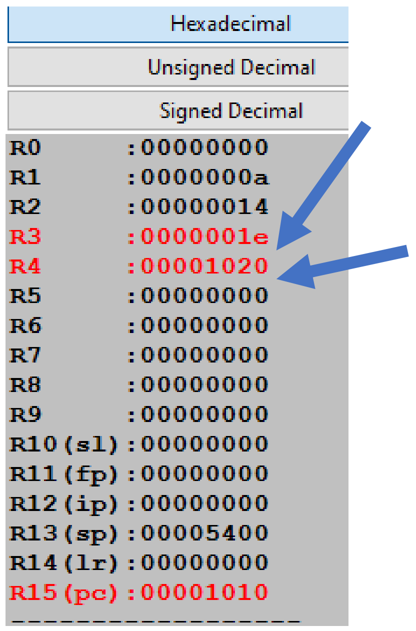
```
R0      :00000000
R1      :00000000
R2      :00000000
R3      :00000000
R4      :00001014
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00005400
R14(lr) :00000000
R15(pc) :00001004
```



```
R0      :00000000
R1      :0000000a
R2      :00000000
R3      :00000000
R4      :00001018
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00005400
R14(lr) :00000000
R15(pc) :00001008
------------------
```



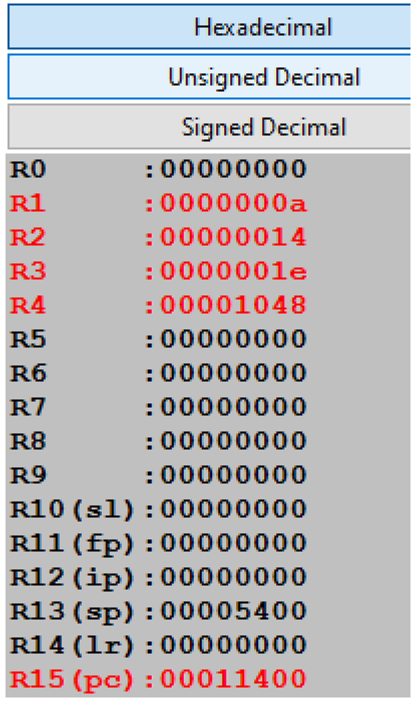
```
R0      :00000000
R1      :0000000a
R2      :00000014
R3      :00000000
R4      :0000101c
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00005400
R14(lr) :00000000
R15(pc) :0000100c
------------------
```



```
R0      :00000000
R1      :0000000a
R2      :00000014
R3      :0000001e
R4      :00001020
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00005400
R14(lr) :00000000
R15(pc) :00001010
------------------
```

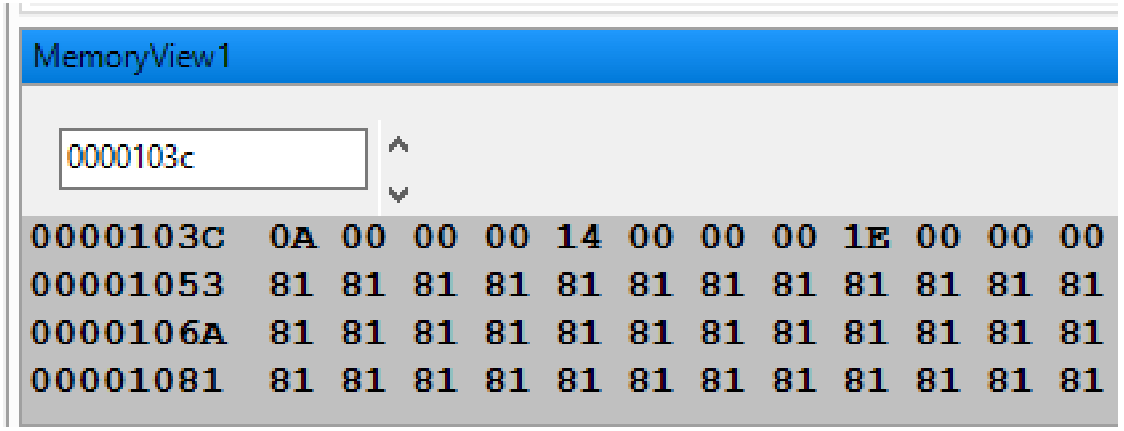
Microprocessor & Computer Architecture (μpCA)

Single Register Load and Store



```
R0      :00000000
R1      :0000000a
R2      :00000014
R3      :0000001e
R4      :00001048
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00005400
R14(lr) :00000000
R15(pc) :00011400
```

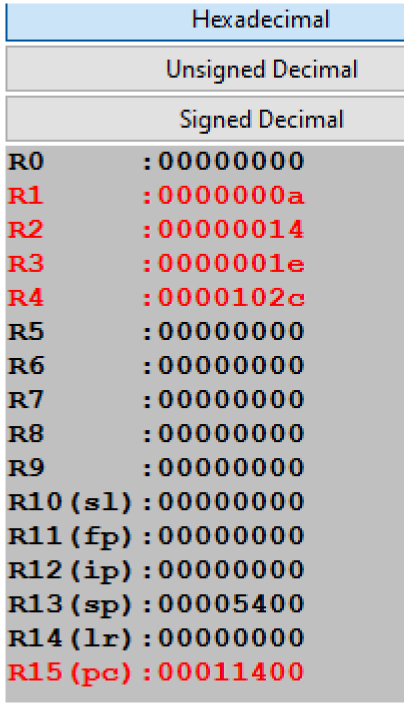
```
.text
LDR R4,=A
LDR R1,[R4],#4
LDR R2,[R4],#4
LDR R3,[R4],#4
LDR R4,=B
STR R1,[R4],#4
STR R2,[R4],#4
STR R3,[R4],#4
.data
A:.word 10,20,30,40,50
000103C    B:.word
```



```
0000103C  0A 00 00 00 14 00 00 00 1E 00 00 00
00001053  81 81 81 81 81 81 81 81 81 81 81 81
0000106A  81 81 81 81 81 81 81 81 81 81 81 81
00001081  81 81 81 81 81 81 81 81 81 81 81 81
```

Microprocessor & Computer Architecture (µpCA)

Multiple Register Load and Store



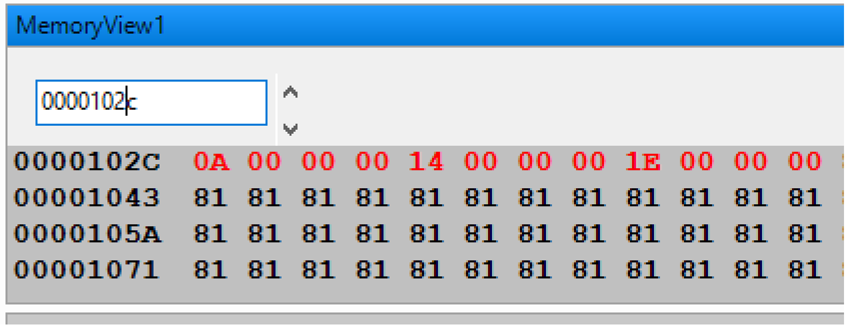
Hexadecimal

Unsigned Decimal

Signed Decimal

```
R0      :00000000
R1      :0000000a
R2      :00000014
R3      :0000001e
R4      :0000102c
R5      :00000000
R6      :00000000
R7      :00000000
R8      :00000000
R9      :00000000
R10(sl) :00000000
R11(fp) :00000000
R12(ip) :00000000
R13(sp) :00005400
R14(lr) :00000000
R15(pc) :00011400
```

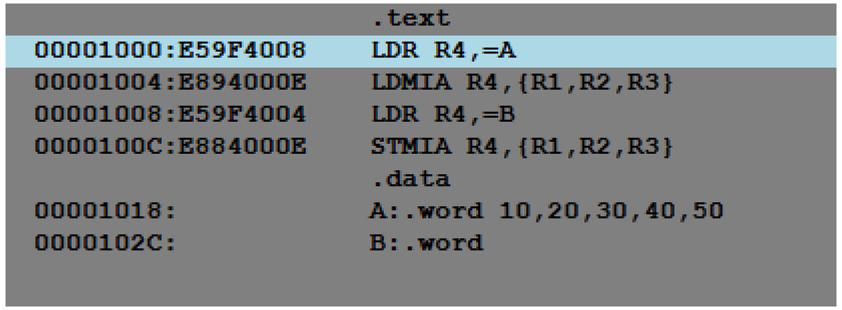
```
        .text
        LDR R4,=A
        LDMIA R4,{R1,R2,R3}
        LDR R4,=B
        STMIA R4,{R1,R2,R3}
        .data
        A:.word 10,20,30,40,50
000102C B:.word
```



MemoryView1

0000102c

```
0000102C  0A 00 00 00 14 00 00 00 1E 00 00 00
00001043  81 81 81 81 81 81 81 81 81 81 81 81
0000105A  81 81 81 81 81 81 81 81 81 81 81 81
00001071  81 81 81 81 81 81 81 81 81 81 81 81
```



```
                         .text
00001000:E59F4008        LDR R4,=A
00001004:E894000E        LDMIA R4,{R1,R2,R3}
00001008:E59F4004        LDR R4,=B
0000100C:E884000E        STMIA R4,{R1,R2,R3}
                         .data
00001018:                A:.word 10,20,30,40,50
0000102C:                B:.word
```

Multiple Register Load and Store or Block Transfer

LDM: **Load multiple registers**

STM: **Store multiple registers**

Syntax:

<LDM/STM> {cond} <Addressing Mode>Rn {!},Registers

Specifying Addressing Mode

Syntax:

<LDM/STM> {cond} <Addressing Mode>Rn {!},Registers

Addressing Mode	Meaning
IA	Increase after
IB	Increase before
DA	Decrease after
DB	Decrease before

Microprocessor & Computer Architecture (µpCA)

Specifying Registers

Syntax:

<LDM/STM> {cond} <Addressing Mode>Rn {!},Registers

LDM <IA/IB/DA/DB> Rn, {R1,R2,R3}

or

LDMIA <IA/IB/DA/DB> Rn, {R1-R3}

Microprocessor & Computer Architecture (μpCA)

LDMIA

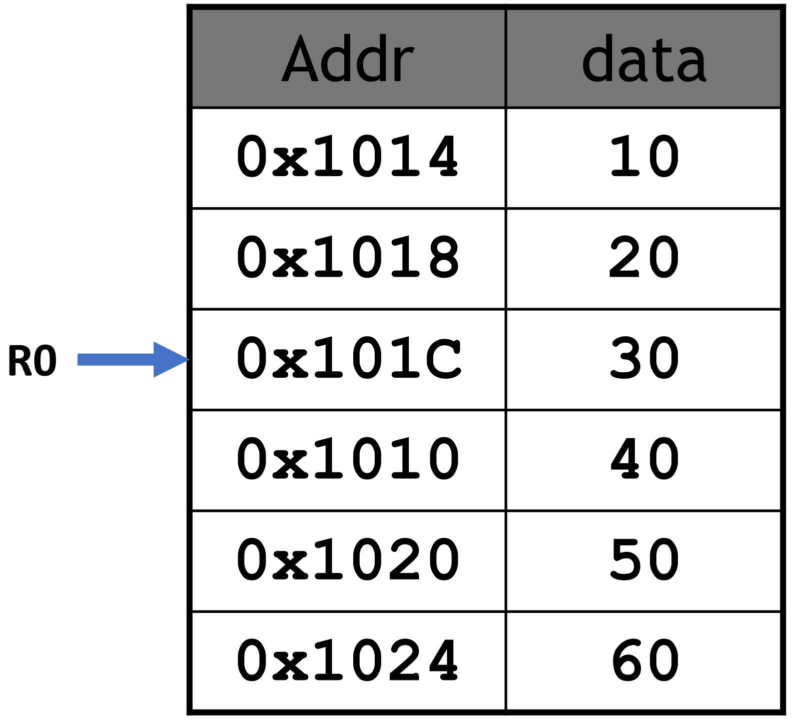
LDM**IA** Rn, {Ri,Ri+1,Ri+2….}
Let Rn=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIA R0, {R1,R2,R3}
or
LDMIA R0, {R1-R3}

R1=30
R2=40
R3=50



Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMIA

LDM**IA** Rn**!**, {Ri,Ri+1,Ri+2….}
Initially R0=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIA R0!, {R1,R2,R3}
or
LDMIA R0!, {R1-R3}

R1=?
R2=?
R3=?

R0

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

LDMIA

LDM**IA** Rn**!**, {Ri,Ri+1,Ri+2….}
Initially R0=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIA R0!, {R1,R2,R3}
or
LDMIA R0!, {R1-R3}

R1=30   Copy First
R2=?
R3=?

Increment After

R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMIA

LDMIA Rn!, {Ri,Ri+1,Ri+2….}
Initially R0=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIA R0!, {R1,R2,R3}
or
LDMIA R0!, {R1-R3}

R1=30
R2=40
R3=?

Copy First

Increment After

R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMIA

LDM**IA** Rn**!**, {Ri,Ri+1,Ri+2….}
Initially R0=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIA R0!, {R1,R2,R3}
or
LDMIA R0!, {R1-R3}

R1=30
R2=40
R3=50
Copy First

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Increment After
R0 →

LDMIB

LDM**IB** Rn**!**, {Ri,Ri+1,Ri+2….}

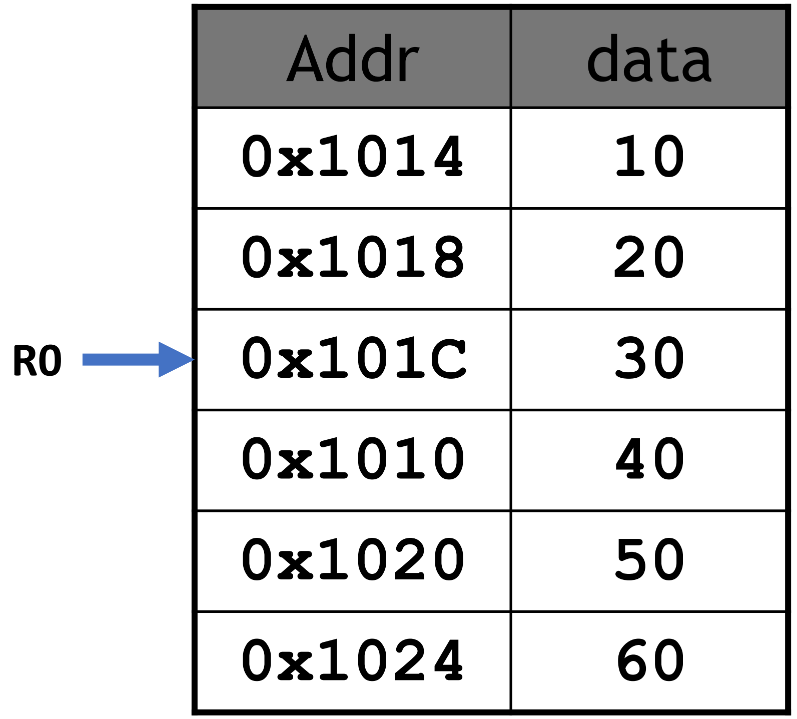
Initially Rn=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIB R0!, {R1,R2,R3}
or
LDMIB R0!, {R1-R3}

R1=?
R2=?
R3=?



Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMIB

LDM**IB** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIB R0!, {R1,R2,R3}
or
LDMIB R0!, {R1-R3}

R1=40 Copy After
R2=?
R3=?

Increment First
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMIB

LDM**IB** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIB R0!, {R1,R2,R3}
or
LDMIB R0!, {R1-R3}

R1=40
R2=50 Copy After
R3=?

Increment First
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMIB

LDM**IB** Rn!, {Ri,Ri+1,Ri+2….}

Initially Rn=0x010C

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMIB R0!, {R1,R2,R3}
or
LDMIB R0!, {R1-R3}

R1=40
R2=50
R3=60
Copy After

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Increment First
R0 →

Microprocessor & Computer Architecture (μpCA)

LDMDA

LDM**DA** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDA R0!, {R1,R2,R3}
or
LDMDA R0!, {R1-R3}

R1=?
R2=?
R3=?

	Addr	data
	0x1014	10
	0x1018	20
	0x101C	30
R0 →	0x1010	40
	0x1020	50
	0x1024	60

Microprocessor & Computer Architecture (µpCA)

LDMDA

LDM**DA** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDA R0!, {R1,R2,R3}
or
LDMDA R0!, {R1-R3}

R1=40 Copy First
R2=
R3=

Decrement After
R0 → 0x1010

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (µpCA)

LDMDA

LDM**DA** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDA R0!, {R1,R2,R3}
or
LDMDA R0!, {R1-R3}

R1=40
R2=30 Copy First
R3=

Decrement After
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (µpCA)

LDMDA

LDM**DA** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDA R0!, {R1,R2,R3}
or
LDMDA R0!, {R1-R3}

R1=40
R2=30
R3=20 Copy First

Decrement After
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMDB

LDM**DB** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDB R0!, {R1,R2,R3}
or
LDMDB R0!, {R1-R3}

R1=?
R2=?
R3=?

R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMDB

LDM**DB** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDB R0!, {R1,R2,R3}
or
LDMDB R0!, {R1-R3}

R1=30 Copy After
R2=
R3=

Decrement First
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

LDMDB

LDM**DB** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDB R0!, {R1,R2,R3}
or
LDMDB R0!, {R1-R3}

R1=30
R2=20 Copy After
R3=

Decrement First
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (µpCA)

LDMDB

LDM**DB** Rn**!**, {Ri,Ri+1,Ri+2….}

Initially Rn=0x1010

Ri =[Rn]
Ri+1=[Rn+4]
Ri+2=[Rn+8]
…………………

Example:

LDMDB R0!, {R1,R2,R3}
or
LDMDB R0!, {R1-R3}

R1=30
R2=20
R3=10 Copy First

Decrement First
R0 →

Addr	data
0x1014	10
0x1018	20
0x101C	30
0x1010	40
0x1020	50
0x1024	60

Microprocessor & Computer Architecture (μpCA)

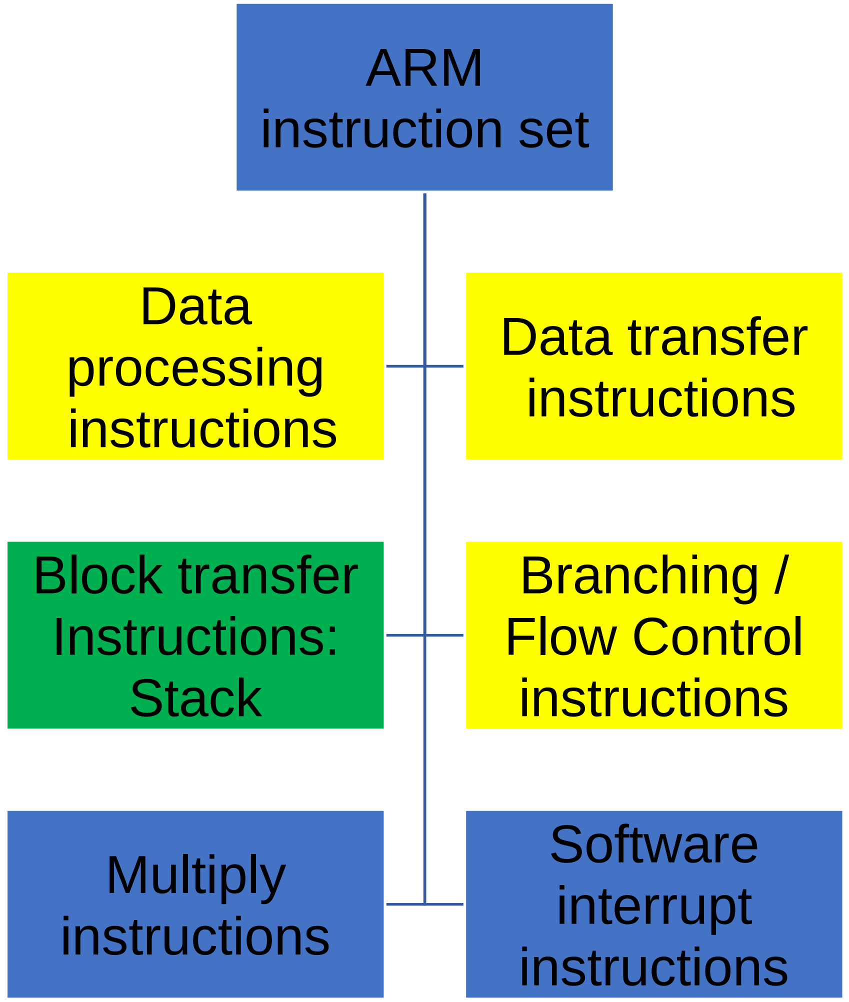
Think and Relate

```
IA:  addr:=Rn

IB:  addr:=Rn+4

DA:  addr:=Rn-#<registers>*4+4

DB:  addr:=Rn-#<registers>*4
```

Addressing mode	Description	Start address	End address	*Rn*!
IA	increment after	Rn	$Rn + 4^*N - 4$	$Rn + 4^*N$
IB	increment before	$Rn + 4$	$Rn + 4^*N$	$Rn + 4^*N$
DA	decrement after	$Rn - 4^*N + 4$	Rn	$Rn - 4^*N$
DB	decrement before	$Rn - 4^*N$	$Rn - 4$	$Rn - 4^*N$

Microprocessor & Computer Architecture (μpCA)

Next Session


ARM instruction set
Data processing instructions
Data transfer instructions
Block transfer Instructions: Stack
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