Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

PES UNIVERSITY ONLINE

Microprocessor & Computer Architecture (µpCA)

Data Transfer Instructions

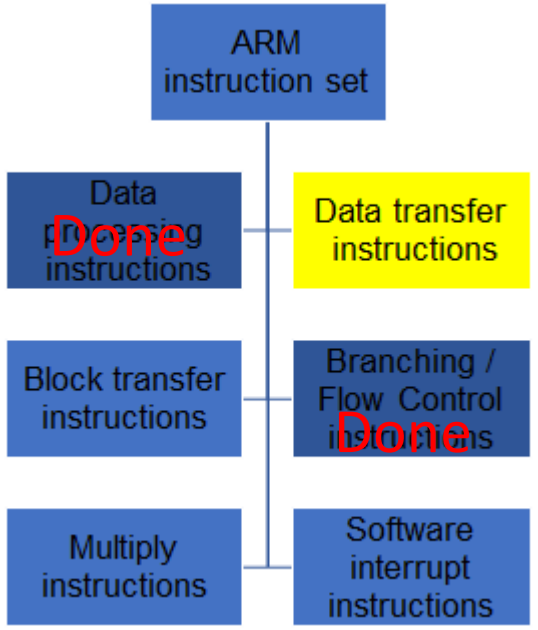
Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Syllabus

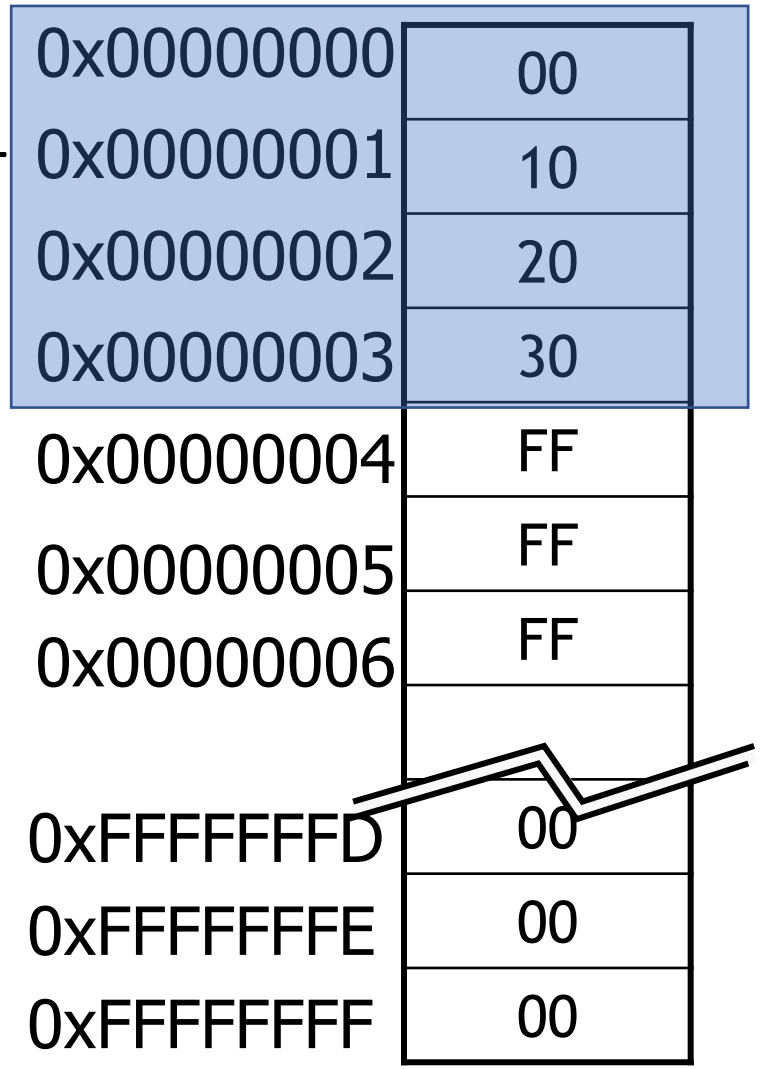
Unit 1: Basic Processor Architecture and Design

- ~~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**



Memory system

- Memory is a linear array of bytes addressed from 0 to 2^{32}-1
- Word, half-word, byte
- Little-endian

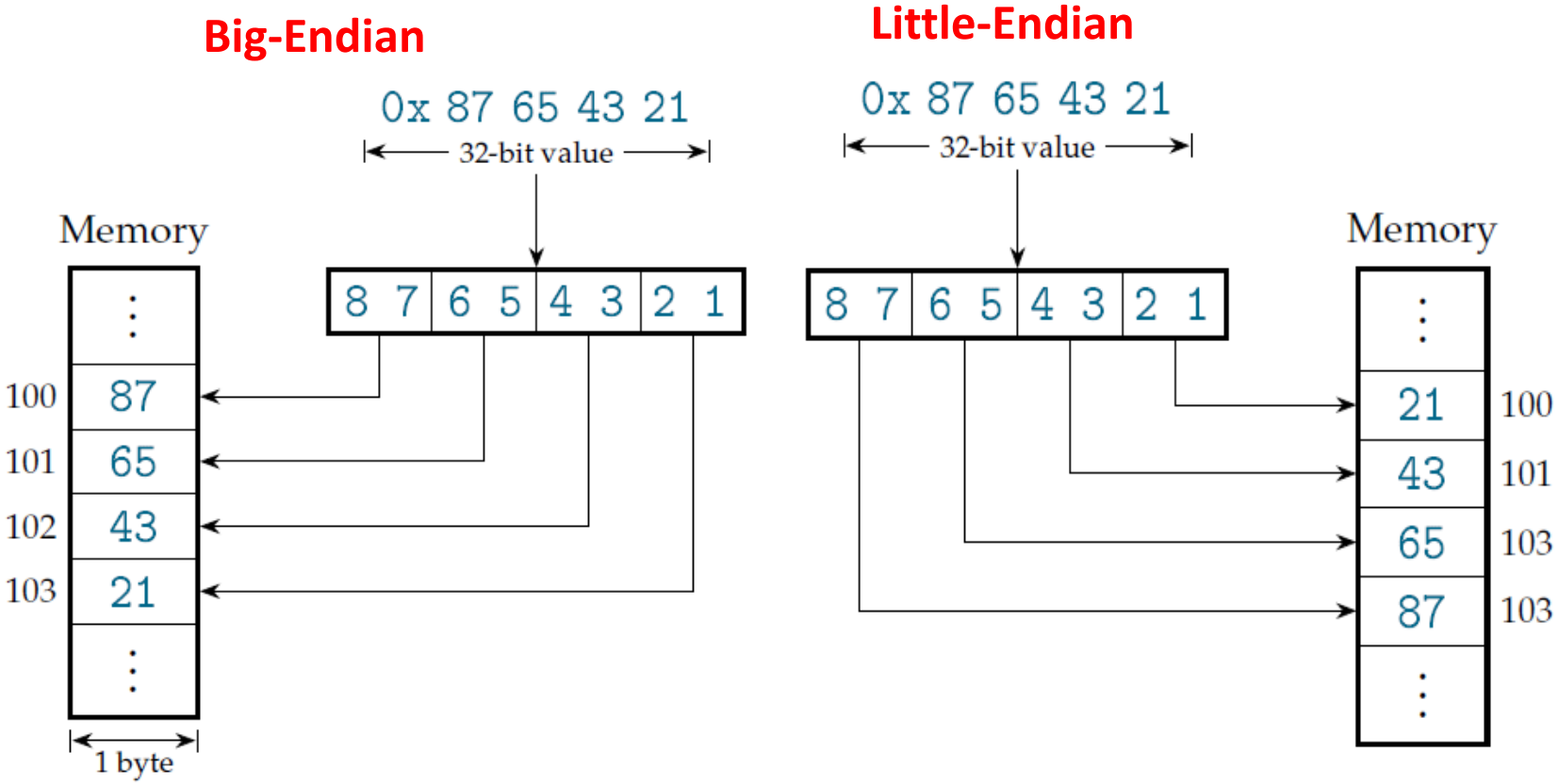


From ARMSIM

```
                        .text
00001000:E3A00013       mov r0,#19
00001004:E59F5018       ldr r5,=a
00001008:E5850000       str r0,[r5]
0000100C:E1A00000       mov r0,r0
00001010:E1A00000       mov r0,r0
00001014:E1A00000       mov r0,r0
00001018:E1A00000       mov r0,r0
0000101C:E1A00000       mov r0,r0
00001020:EF000011       swi 0x011
                        .data
00001028:               a: .word 0
```

Little-Endian vs. Big-Endian

- Little-endian: least-significant byte first
- Big-endian: most-significant byte first



- Move data between registers and memory
- Basic forms
  - Single register load/store
    - Ex: **LDR , STR**
  - Multiple register load/store or Block Transfer
    - Ex: **LDM, STM**

Single Register Load/Store

Syntax: <LDR/STR>{<cond>}{B} Rd, Addressing

LDR	Load word into register
STR	Save byte or word from register
LDRB	Load byte into register
STRB	Save byte from Register

Single Register Load/Store

Syntax: LDR{<cond>}SB/H/SH Rd, Addressing
STR{<cond>}H Rd, Addressing

LDRH	Load half word into register
STRH	Save half word from a register
LDRSB	Load signed byte into register
LDRSH	Load signed halfword into a Register

No **`STRSB/STRSH`** since **`STRB/STRH`** stores both signed/unsigned ones

Copy A=B

	addr	data
	0x010	10
Let B →	0x014	
	0x018	30
Let A →	0x01C	40
	0x020	50
	0x024	60

```
LDR R0=A   ;Where a is a variable i.e address is copied to Register R0= 0x1C

LDR R5, [R0]    ; Copying the data in the address in R0 to R5=40

LDR R3=B   ;Where b is a variable i.e address is copied to Register R3 = 0x14
STR R5,[R3]  ; Store 40 in the address specified in R3=?
```

Copy A=B

	addr	data
	0x010	10
Let B →	0x014	
	0x018	30
Let A →	0x01C	40
	0x020	50
	0x024	60

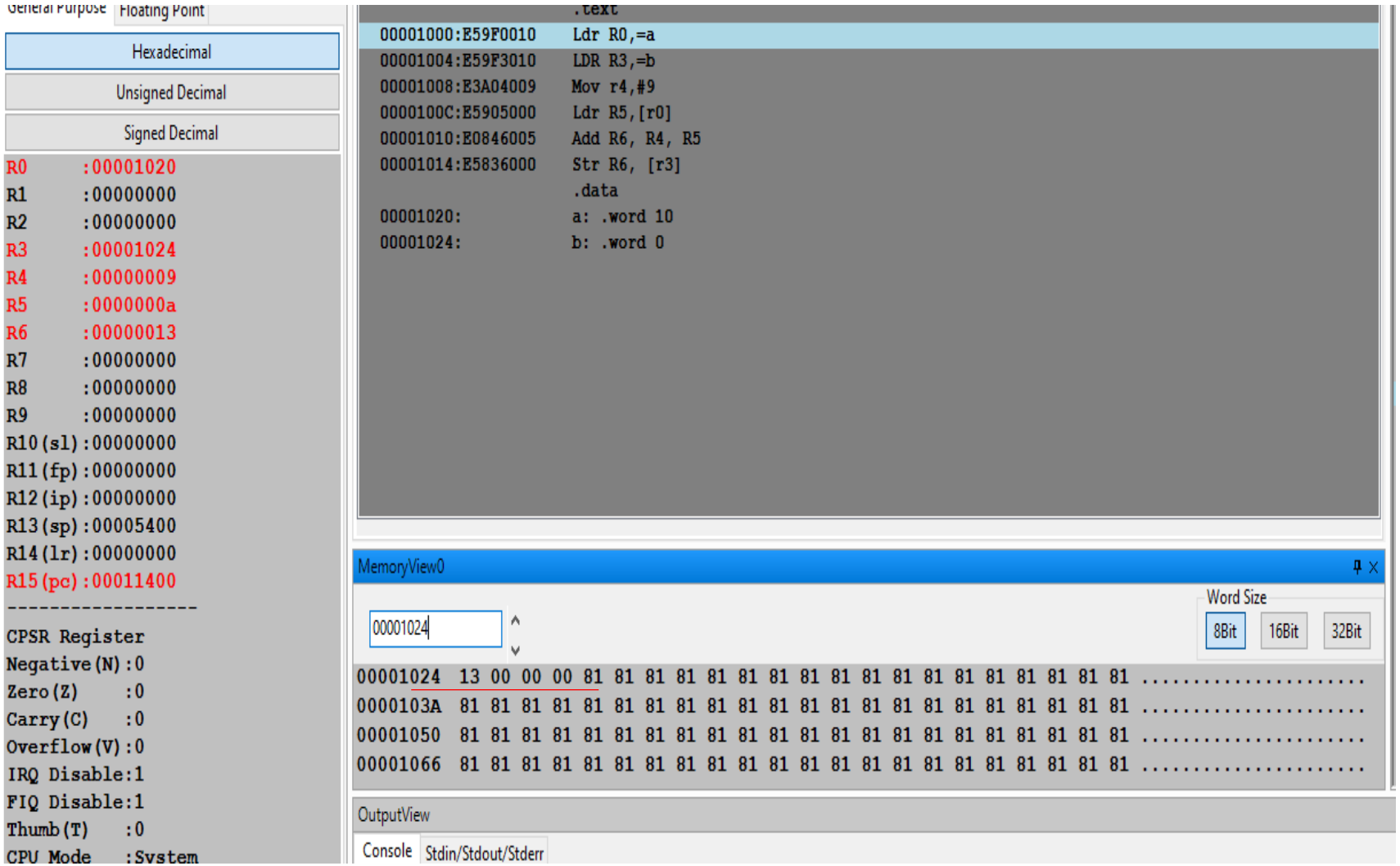
```
LDR R0=A   ;Where a is a variable i.e address is copied to Register R0= 0x1C

LDR R5, [R0]    ; Copying the data in the address in R0 to R5=40

LDR R3=B   ;Where b is a variable i.e address is copied to Register R3 = 0x14
STR R5,[R3]  ; Store 40 in the address specified in R3=40
```

LDR and STR Example B=A+9

```
.text
LDR R0, =A
LDR R3,=B
MOV R4,#9
LDR R5, [R0]
ADD R6, R4, R5
STR R6, [R3]
.data
A: .word 10
B: word 0
```

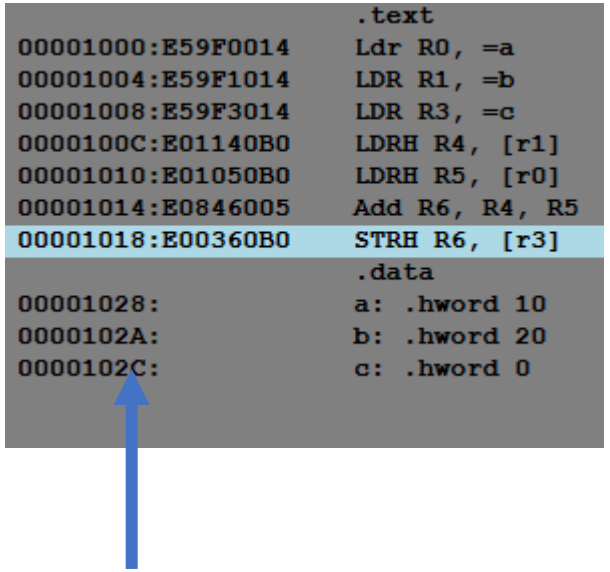
- LDR R0, =A → Address of A is copied to R0
- LDR R3,=B → Address of B is copied to R3
- MOV R4,#9 → Number 9 is copied to R4
- LDR R5, [R0] → Value in the A i.e 10 is copied to R5
- ADD R6, R4, R5 → 10+9=19 copy result to R6
- STR R6, [R3] → 19 is stored in the address in the register [R3], ie. B=19

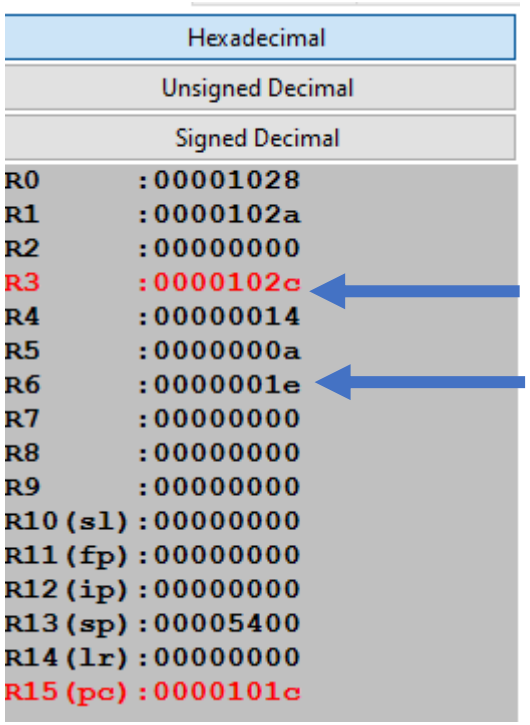
Microprocessor & Computer Architecture (µpCA)

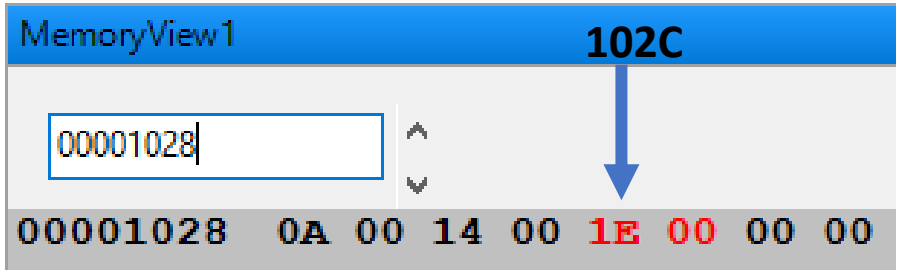


Example 2: C=A+B (Half Word)

```
.text
LDR R0, =a
LDR R1, =b
LDR R3, =c
LDRH R4, [R1]
LDRH R5, [R0]
ADD R6, R4, R5
STRH R6, [R3]
.data
a: .hword 10
b: .hword 20
c: .hword 0
```







Microprocessor & Computer Architecture (µpCA)

Example 3: String (BYTE)

```
.text
Ldr         R0, =mystring
Swi 0x02
Swi 0x11
.data
mystring:.asciz "HELLO PESU"
```

jstring.s

```
                        .text
00001000:E59F0004       Ldr     R0, =mystring
00001004:EF000002       Swi 0x02
00001008:EF000011       Swi 0x11
                        .data
00001010:               mystring:.asciz "HELLO PESU"
```

Hexadecimal
Unsigned Decimal
Signed Decimal

```
R0      :00001010
R1      :00000000
R2      :00000000
R3      :00000000
R4      :00000000
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
----------------
```

MemoryView1

00001010

```
00001010  48 45 4C 4C 4F 20 50 45 53 55 00 00 81
00001027  81 81 81 81 81 81 81 81 81 81 81 81 81
```

OutputView

Console | Stdin/Stdout/Stderr

```
HELLO PESU
```

Addressing or Indexing

- Pre Indexing Without Write Back
- **Syntax:** LDR Rd, [Rn,OFFSET]
- Pre Indexing With Write Back

  Syntax: LDR Rd, [Rn,OFFSET]!
- Poste Indexing

  Syntax: LDR Rd, [Rn] ,OFFSET

THANK YOU

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

dckiran@pes.edu

9829935135