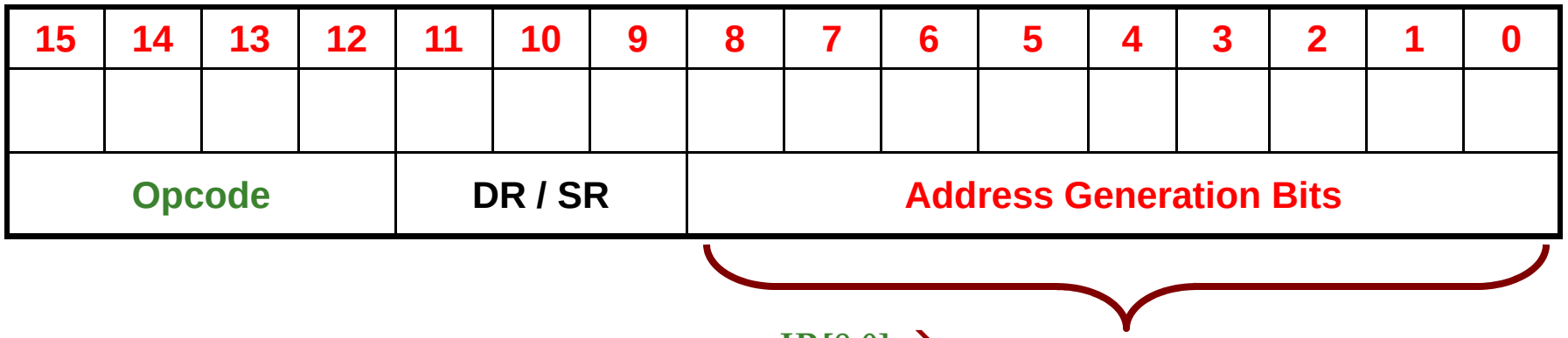
COMPUTER ORGANIZATION (IS F242)

LECT 17: LC 3 ARCHITECTURE

Format of Load / Store Instructions



IR[8:0] →

Used to generate 16 bit address in the memory.

Four ways to interpret these 9 bits, collectively called as *addressing modes*.

Data Movement Instructions

- **Load -- read data from memory to register**
  - **LD: PC-relative mode**
  - **LDR: base+offset mode**
  - **LDI: indirect mode**
- **Store -- write data from register to memory**
  - **ST: PC-relative mode**
  - **STR: base+offset mode**
  - **STI: indirect mode**
- **Load effective address -- compute address, save in register**
  - **LEA: immediate mode**
  - **does not access memory**

PC Relative Mode – Direct Addressing

Want to specify address directly in the instruction

- But an address is 16 bits, and so is an instruction!
- After subtracting 4 bits for opcode
  and 3 bits for register, we have 9 bits available for address (IR[8:0]).

Solution:

- Use the 9 bits as a *signed offset* from the current PC.

9 bits → $-256 \leq \text{offset} \leq +255$

Can form any address X, such that → $PC - 256 \leq X \leq PC + 255$

(from the current instruction -255 to +256)

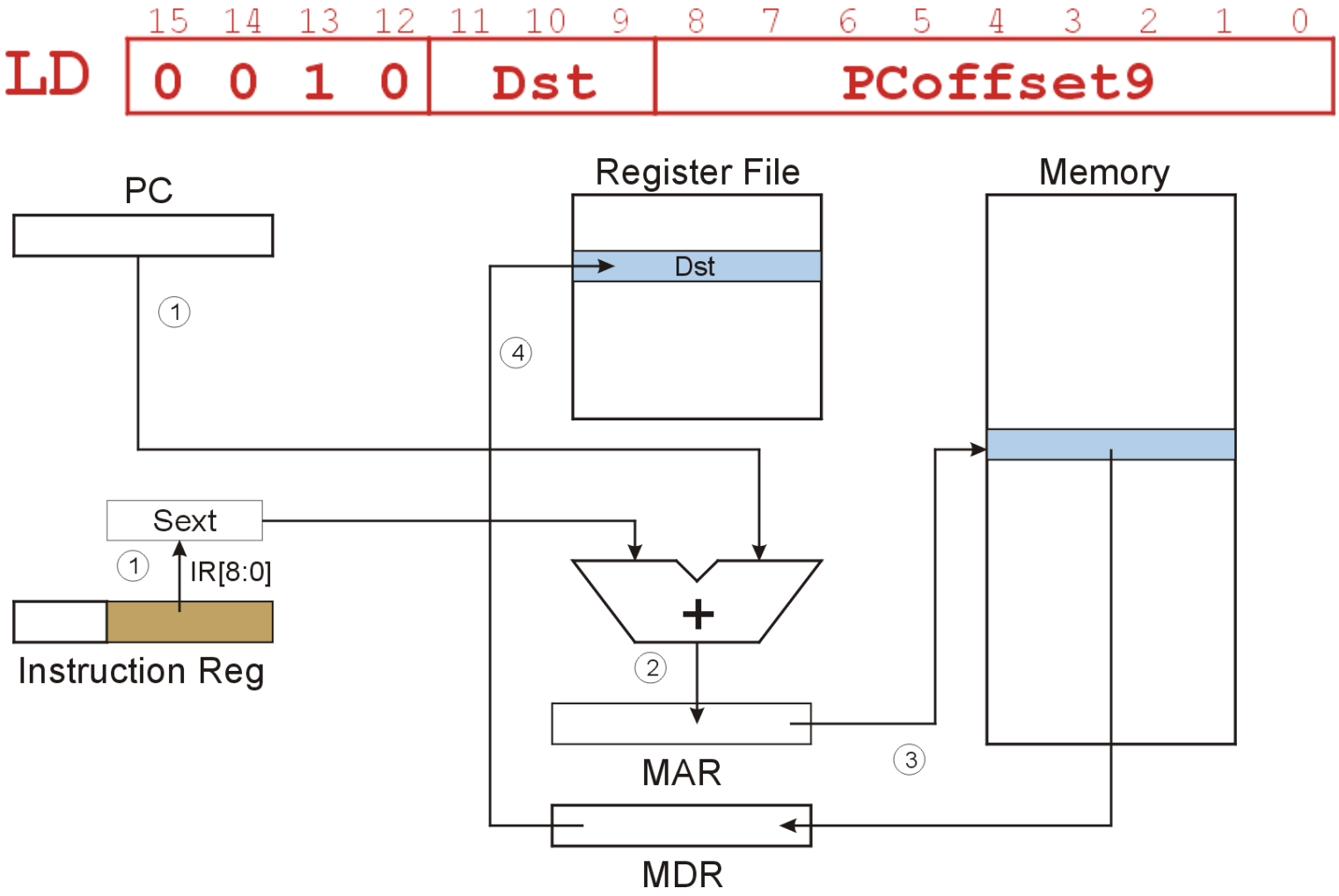
Effective Address = PC + SEXT(IR[8:0])

Remember that PC is incremented as part of the FETCH phase;

- Hence, the incremented PC is used in calculating the address

- Operand location must be within 256 locations of the instruction

PC Relative Mode (Load LD Instruction)



What does the following instruction do ?

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
Opcode LD				Destination Register R2			PC Offset (9 bits) x1AF								

Assume this LD instruction is in the location x2345. ➔ While this instruction is in execution, PC will have x2346.

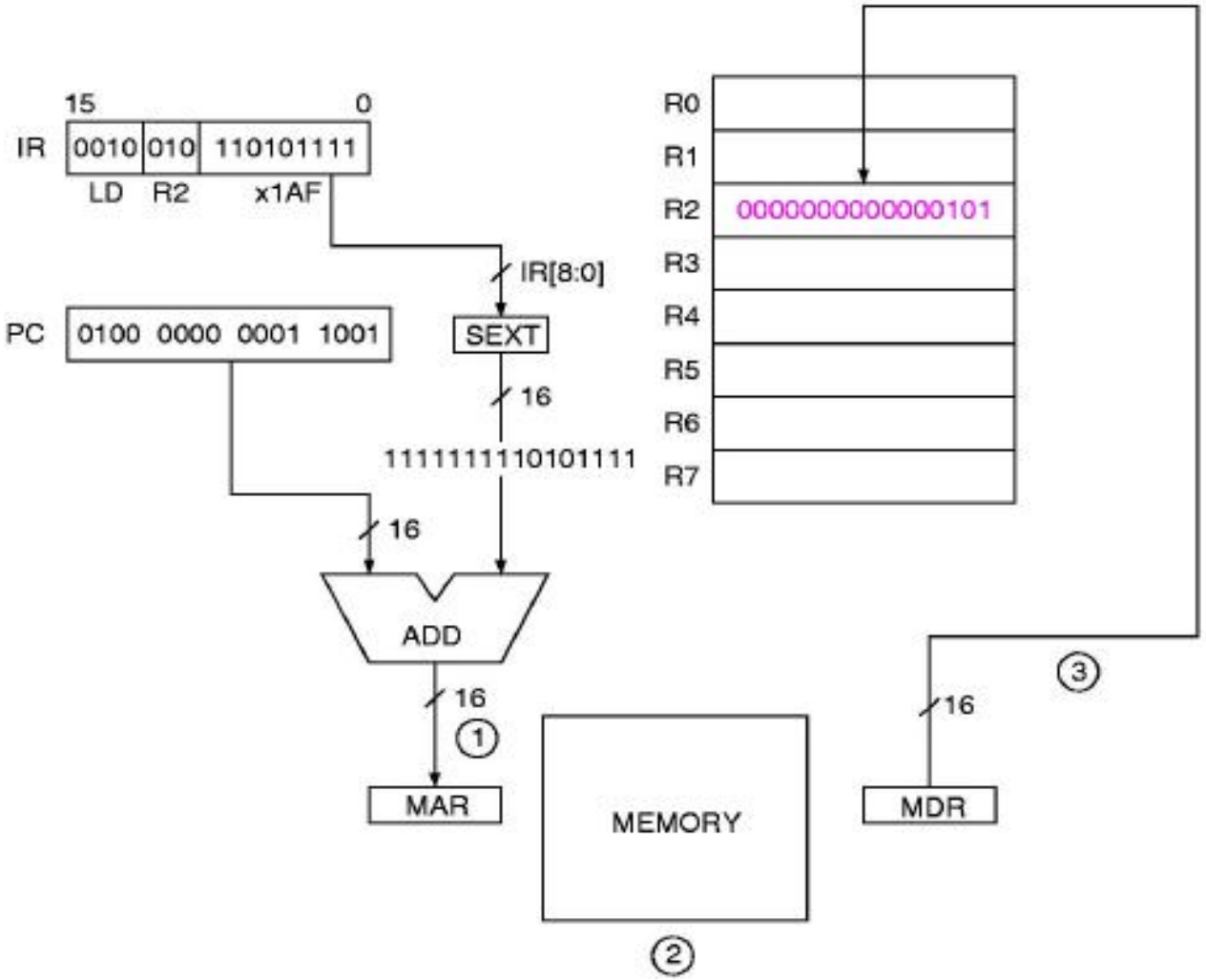
Sign Extended offset is xFFAF

The address for load instruction is now xFFAF + x 2346

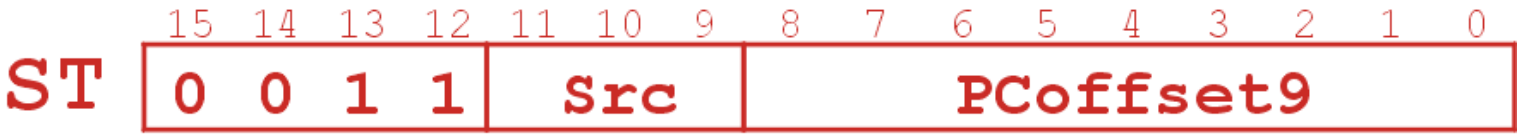
The content of memory location xFFAF + x 2346 will be loaded into R2

LD data path

LD R2, x1AF



ST (PC-Relative)



Base + offset Addressing Mode

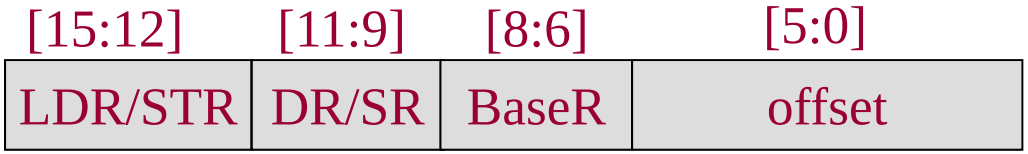
- With PC-relative mode, can only address data within 256 words of the instruction.
  - What about the rest of memory?

- *Solution*
  - Read address from (register + offset) location, then load/store from/to that address.

Base + Offset Mode

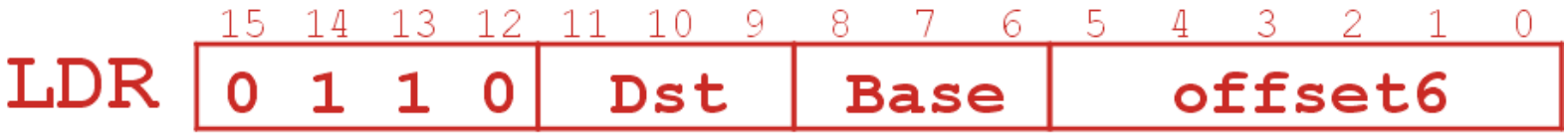
- Relative (Base + Offset) addressing

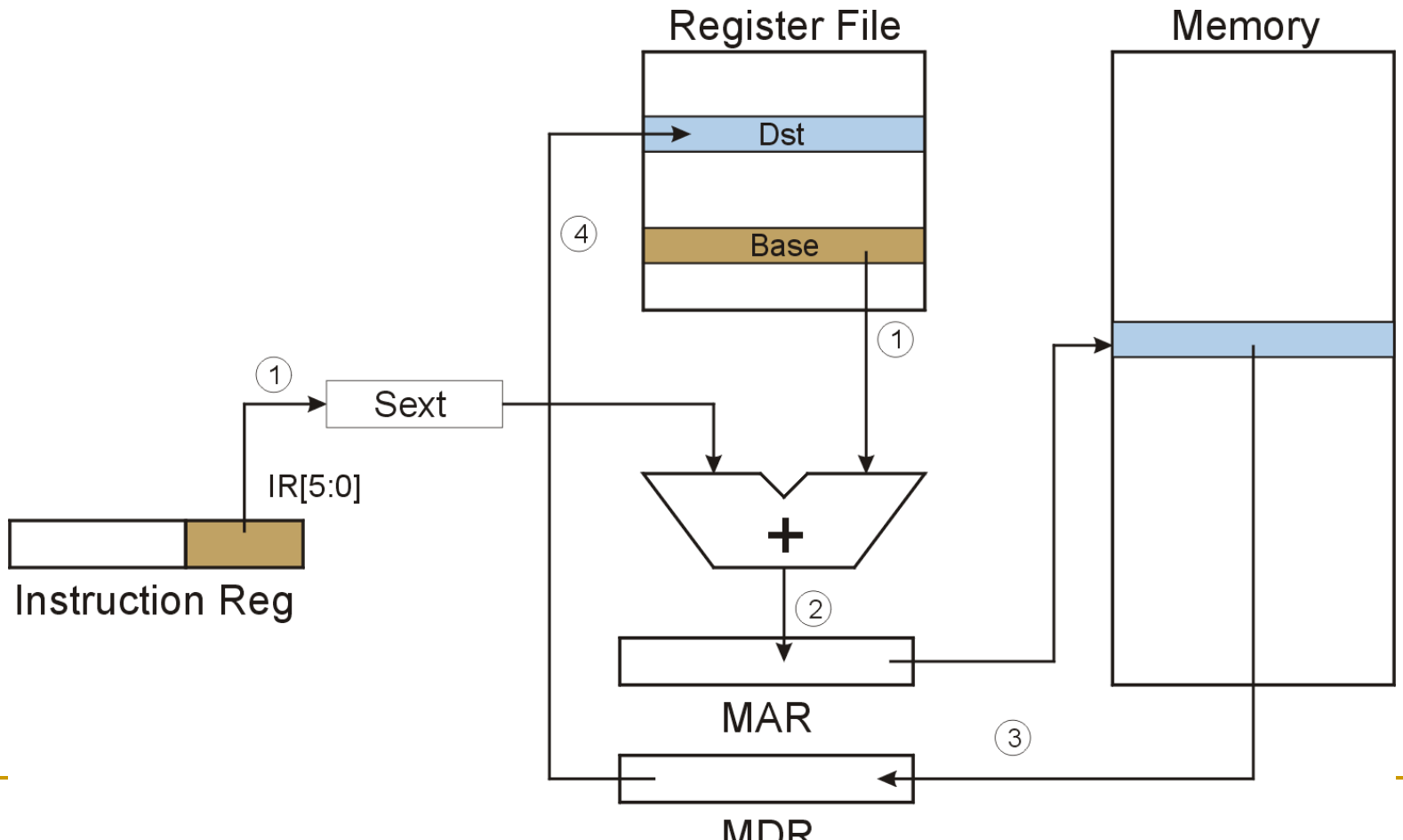
  LDR (opcode 0110) & STR (opcode 0111)



  - effective address = (Base Register) + offset
    - sign extend (SEXT) the 6 bit offset ([5:0]) (-32 to +31) to 16 bits
    - add it to the contents of the Base Register ([8:6])
  - Differences from Direct addressing (PC-Relative):
    - base+offset field is 6 bits, PC-Relative offset field is 9 bits
    - Base + offset can address any location in memory,
    - PC-Relative offset only within +/- 256 locations of the instruction.

LDR (Base+Offset)





What does the following instruction do ?

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1
Opcode LDR				Destination Register R6			Base Register R5			Signed Offset (6 bits) x2F					

Let say R5 contains x6AC9.

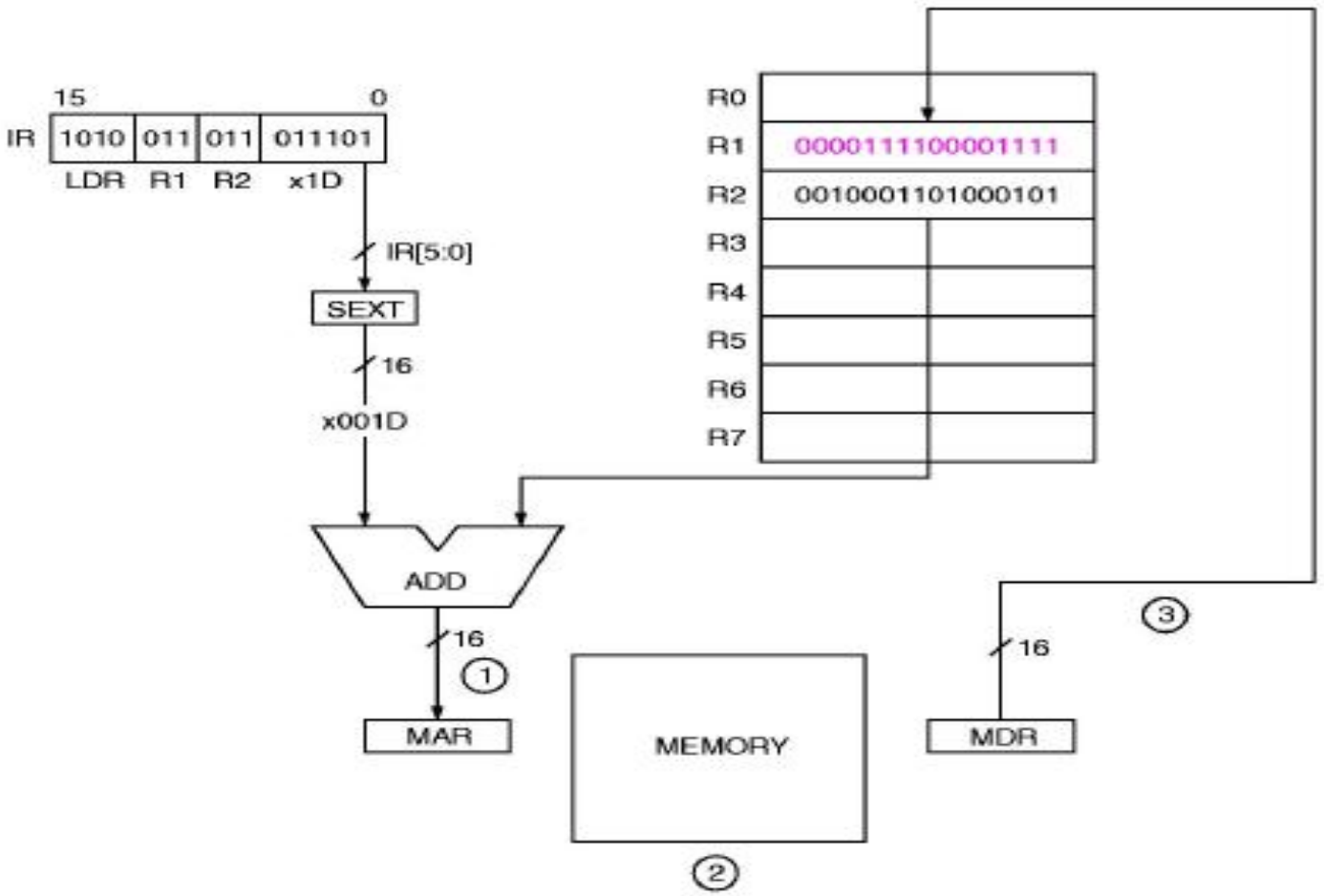
Signed extended offset becomes→ xFFEF

Operand address will be→ x6AC9 + xFFEF

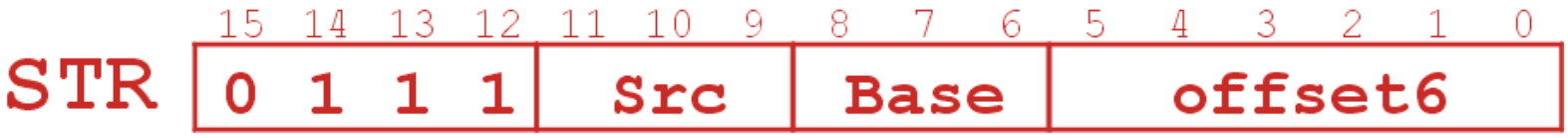
Content of memory location **(x6AC9 + xFFEF)** will be loaded in R6

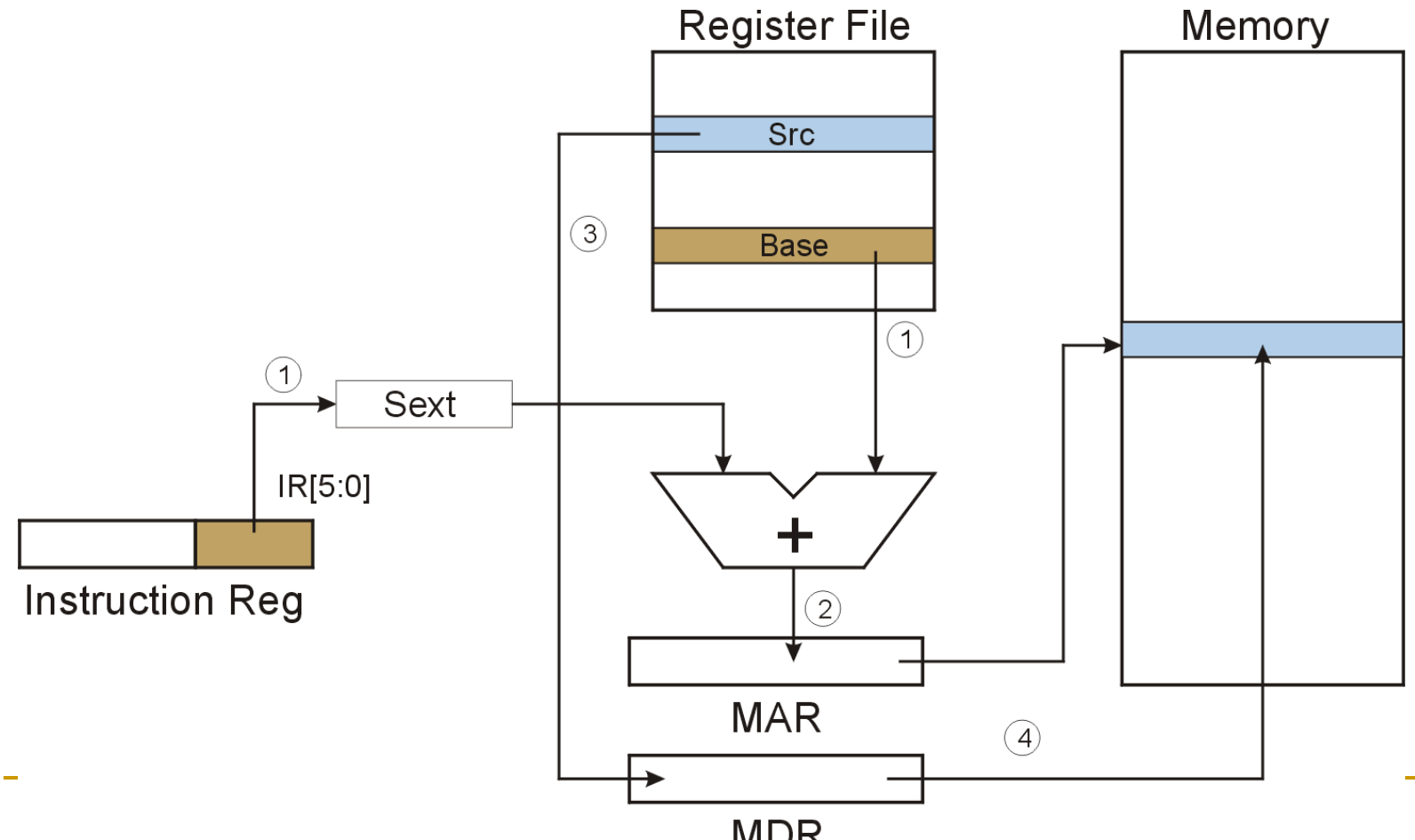
LDR data path

LDR R1, R2, x1D



STR (Base+Offset)





What does the following instruction do ?

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	1	0	1	0	1	1	0	1	1	1	1
Opcode STR				Source Regi R6			Base Register R5			Signed Offset (6 bits) x2F					

Let say R5 contains x6AC9.

Signed extended offset becomes→ xFFEF

Operand address will be→ **x6AC9 + xFFEF**

Contents of R6 will be store in memory location **(x6AC9 + xFFEF)**

Indirect Addressing Mode

- With PC-relative mode, can only address data within 256 words of the instruction.
  - What about the rest of memory?

- *Solution*
  - Read address from memory location, then load/store from/to that address.

- First address is generated from PC and IR (just like PC-relative addressing), then content of that address is used as target for load/store.

Indirect Addressing

- Indirect addressing

  LDI (opcode 1010) & STI (opcode 1011)

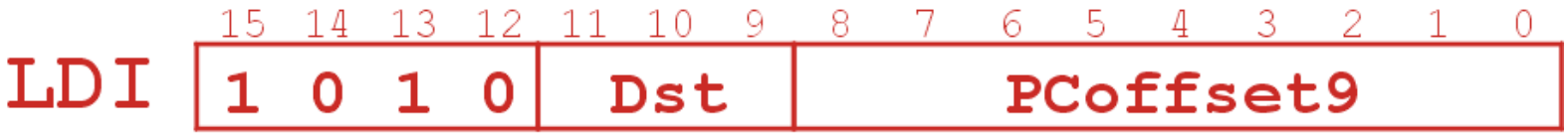
  - Same initial mechanism as direct mode (i.e. PC-Relative), but the calculated memory location now contains the address of the operand, (i.e. the addressing is indirect):

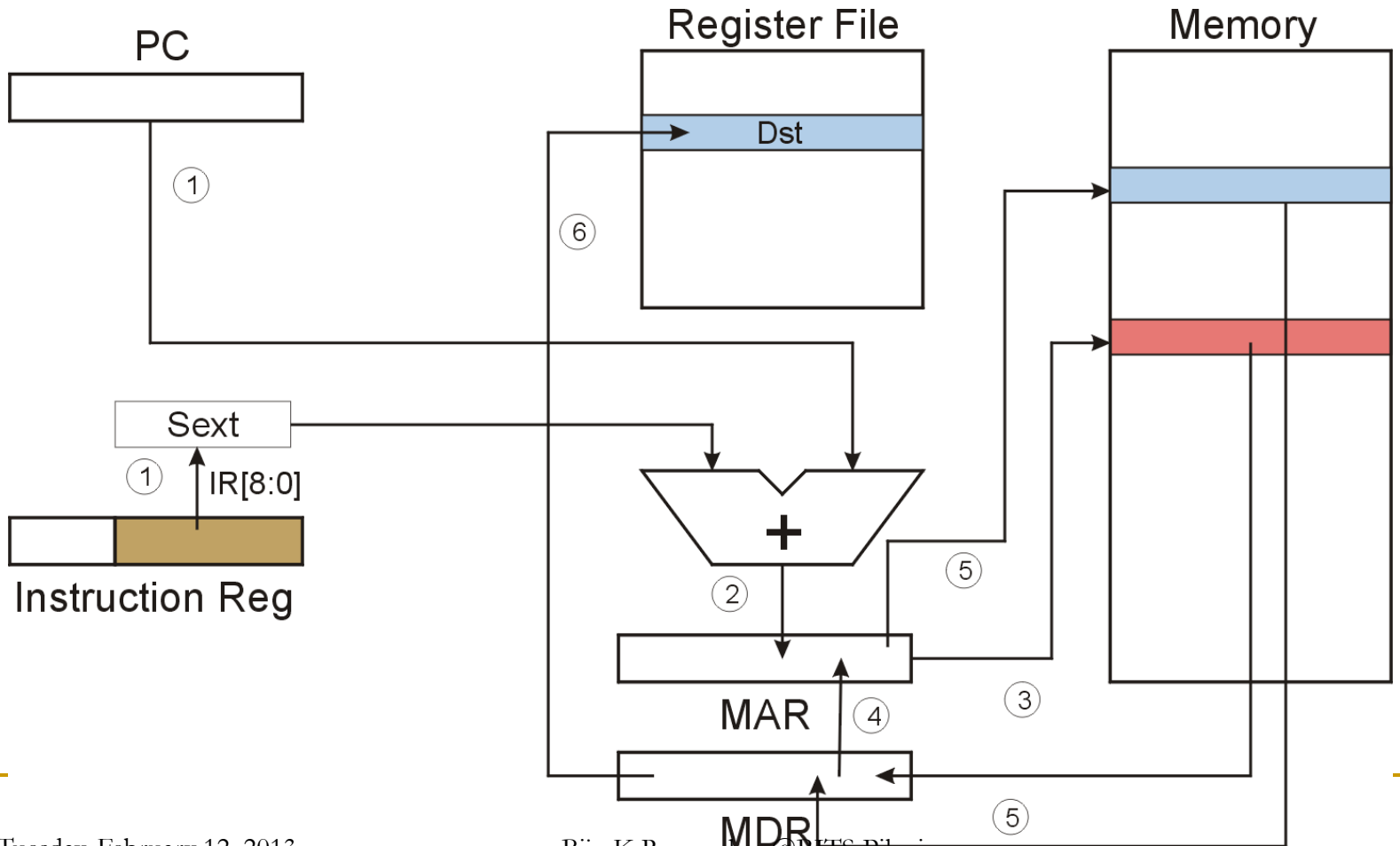
    pointer address = (PC) + SEXT(IR[8:0])

    effective address = Mem [(PC) + SEXT(IR[8:0])]

    - Memory has to be accessed twice to get the actual operand.
    - Address can be anywhere in Memory

LDI (Indirect)





What does the following instruction do ?

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	0	1	1	0	1	0	1	1	1	1
Opcode LDI				Destination Register R6			PC Offset (9 bits) x1AF								

Assume this LDI instruction is in the location x2345. ➔ While this instruction is in execution, PC will have x2346.

Sign Extended offset is xFFAF

The address of operand's address is → xFFAF + x 2346

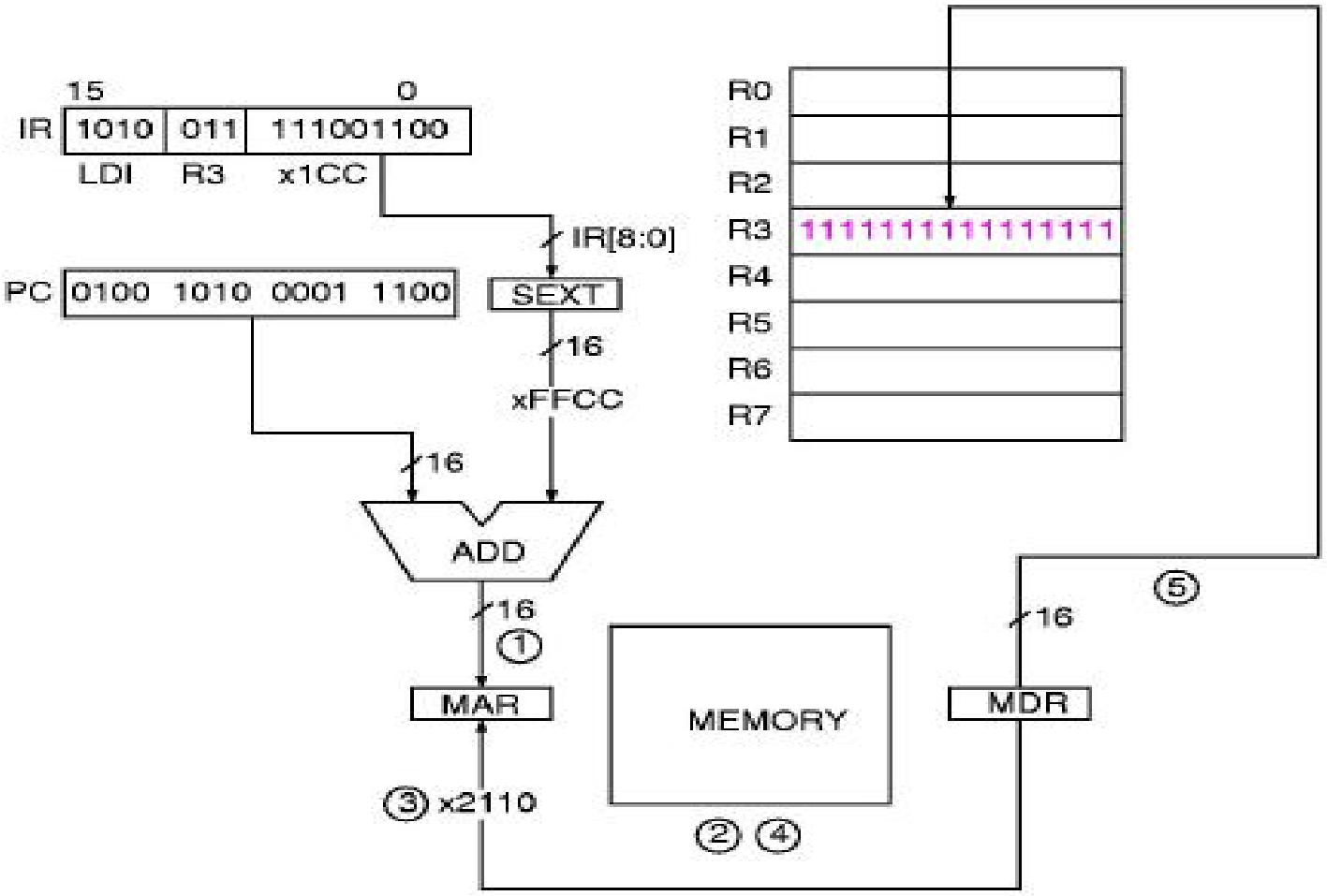
Assume the contents of the memory location (xFFAF + x 2346)→ xABC6

So the address of the operand becomes→ xABC6

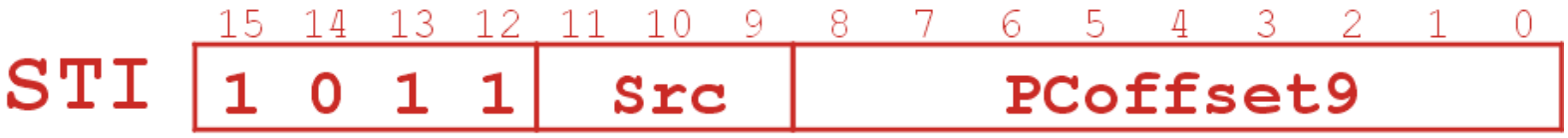
So finally contents of memory location xABC6 will be loaded in R6

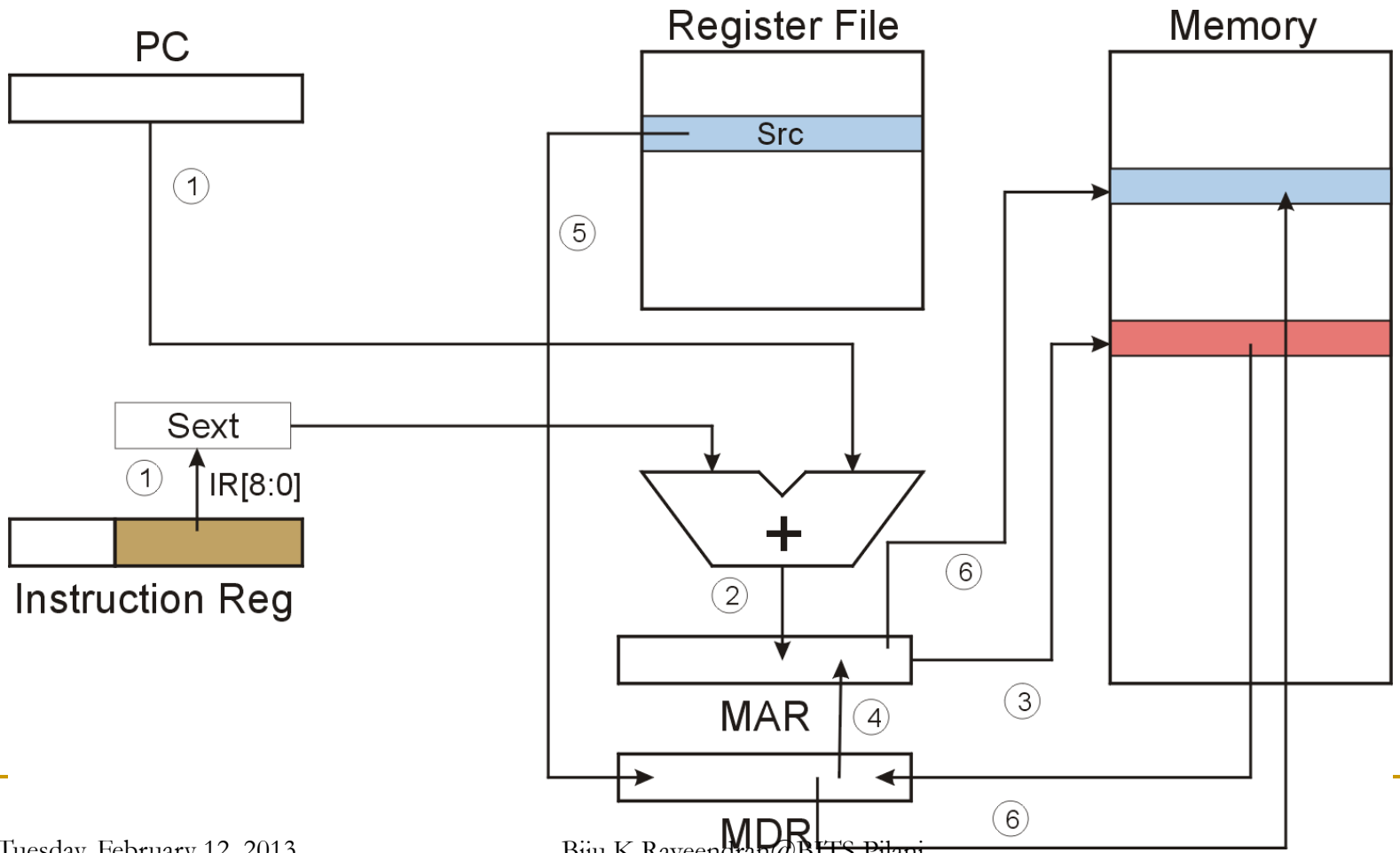
LDI data path

LDI R3, x1CC



STI (Indirect)





What does the following instruction do ?

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	1	1	0	1	0	1	1	1	1
Opcode STI				Destina... Regis R6			PC Offset (9 bits) x1AF								

Assume this STI instruction is in the location x2345. ➔ While this instruction is in execution, PC will have x2346.

Sign Extended offset is xFFAF

The address of operand's address is → xFFAF + x 2346

Assume the contents of the memory location (xFFAF + x 2346)→ xABC6

So the address of the operand becomes→ xABC6

So finally contents of R6 will be STORED in memory location xABC6

Load Effective Address (Immediate)

- Computes address like PC-relative -> PC + signed offset and stores the result into a register

- The *address* is stored in the register, not the contents of the memory location.

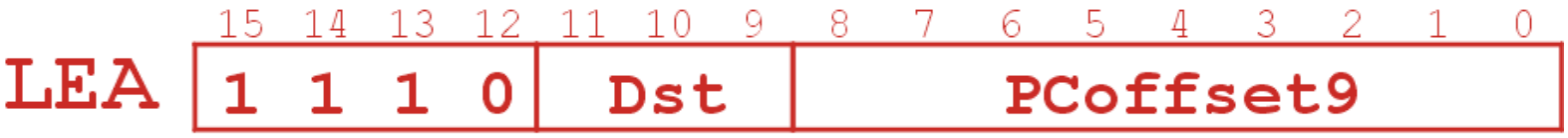
- Why Immediate?
  - No memory access required for operand loading

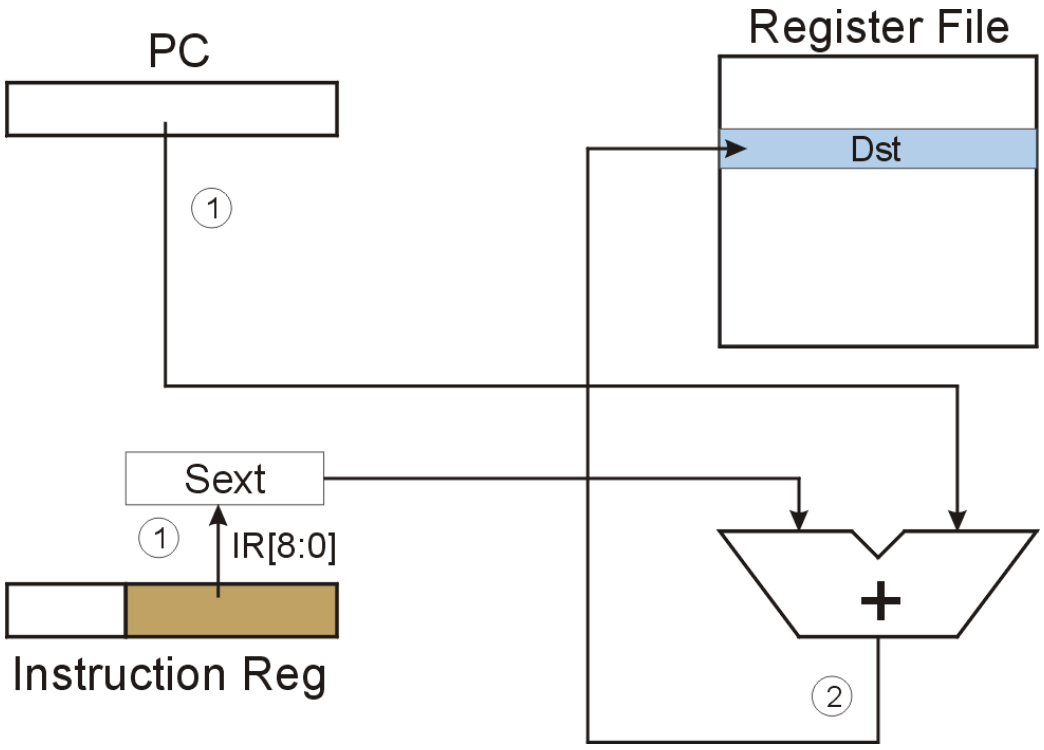
Immediate Mode

- LEA (opcode 1110)

- Loads the register specified by bits [11:9] by the value formed by adding the incremented PC to the sign-extended bits [8:0] of the instruction.
- DR <= (PC) + SEXT(IR[8:0])
- Note : Does not access memory.
- Useful to initialize a register with an address that is very close to the address of the instruction doing the initialization

LEA


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
LEA
1 1 1 0
Dst
PCoffset9


PC
Register File
Dst
1
Sext
1
IR[8:0]
+
Instruction Reg
2

LEA data path

LEA R5, # -3