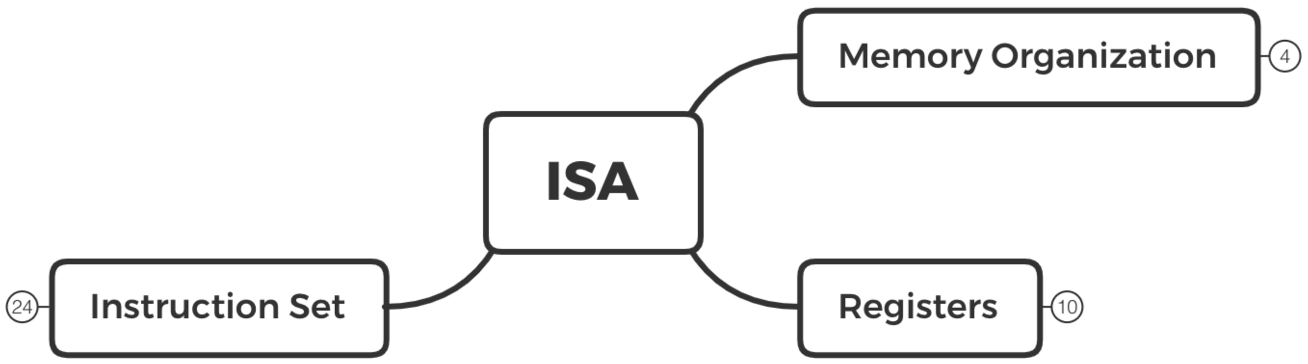
Patt-Ch5 The LC-3 ISA & Data Path

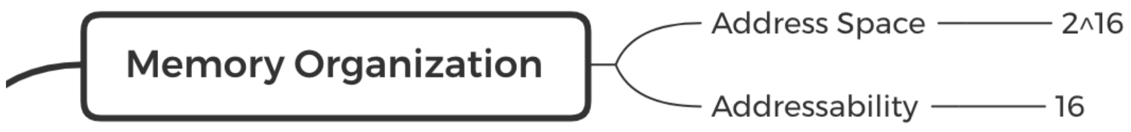
5.1 The ISA: Overview

ISA = All of the programmer-visible components and operations of the computer

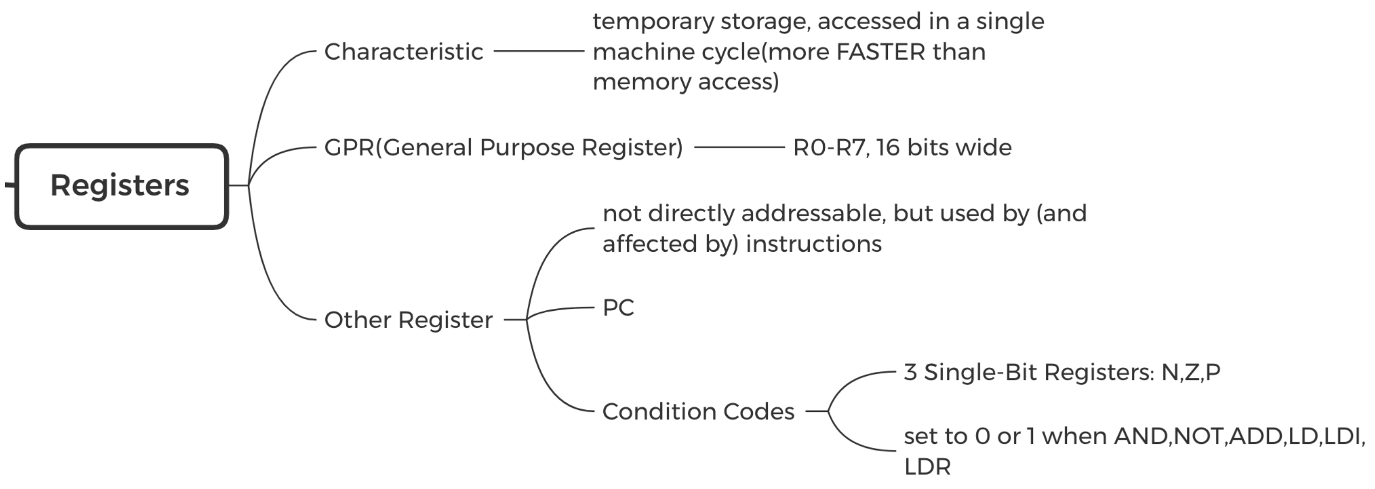
- ISA provides all information needed for someone that wants to write a program in **machine language**
- (or translate from a high-level language to machine language, that is **compiler** and **assembler**)



5.1.1 Memory Organization



5.1.2 Registers


Registers
Characteristic
temporary storage, accessed in a single machine cycle(more FASTER than memory access)
GPR(General Purpose Register)
R0-R7, 16 bits wide
Other Register
not directly addressable, but used by (and affected by) instructions
PC
Condition Codes
3 Single-Bit Registers: N,Z,P
set to 0 or 1 when AND,NOT,ADD,LD,LDI, LDR

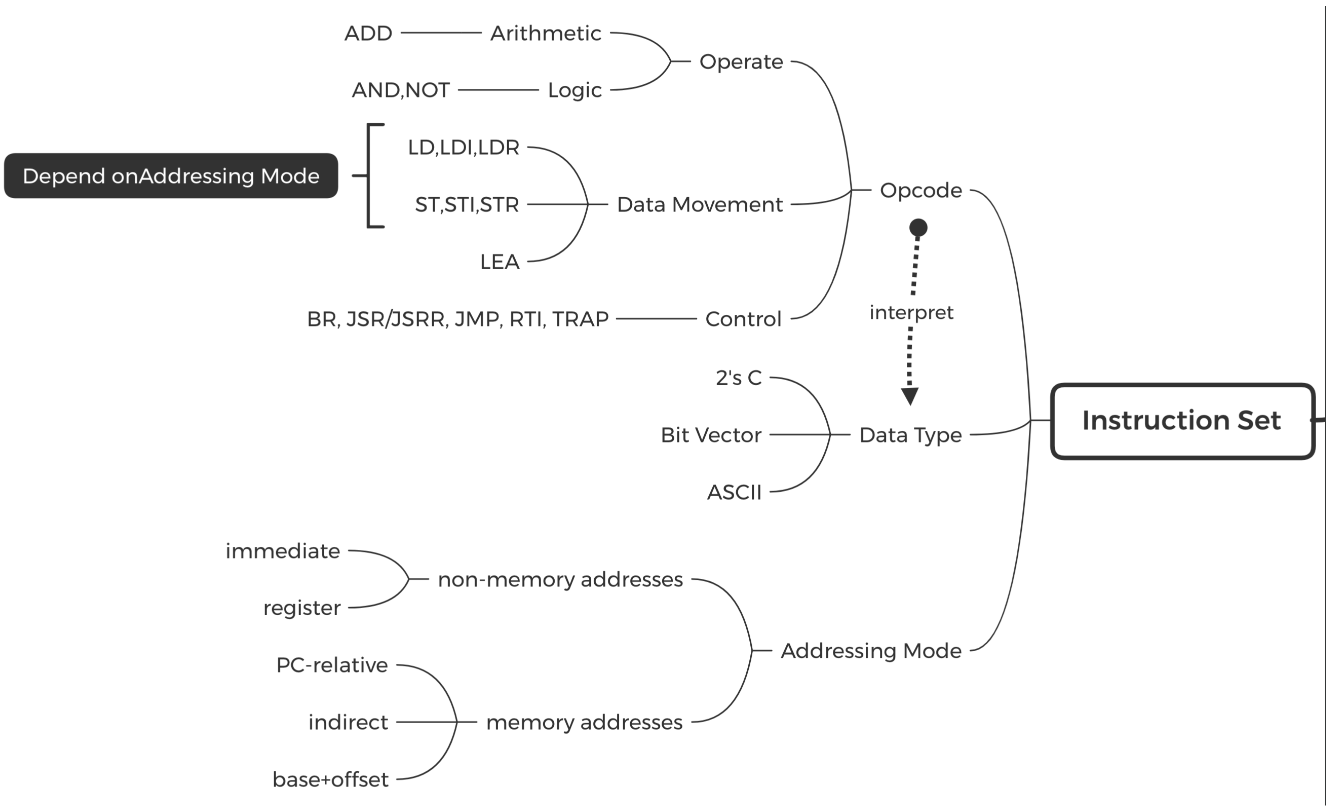
Condition Codes

- set or cleared each time one of the eight GPRs is written into as a result when perform **ADD,AND,NOT,LD,LDI,LDR**
- corresponding to whether the result written to the GPR is negative, zero, or positive.
  - If the result is negative, the N register is set, and Z and P are cleared.
  - If the result is zero, Z is set and N and P are cleared.
  - If the result is positive, P is set and N and Z are cleared.

*Logic Circuit of **P** Register*

> Actually, the graph is totally wrong. Because it is not *opcode* that serves as WE, but the *state.*

5.1.3 The Instruction Set



Addressing Mode

- Machanism for specifying where to fetch the operand

> Instruction Display: machine code, dataflow diagram, data path(save for 5.5)

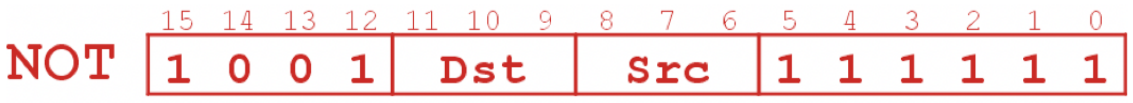
5.2 Operate Instructions

- Opcode: ADD, AND, NOT
- Data Type: 2's C for ADD, bit vector for LOGIC
- Addressing Mode: imm, register, *not including memory*

5.2.1 NOT

> the only *unary* operation

machine code format



- Src and Dst can be the ***same***

dataflow

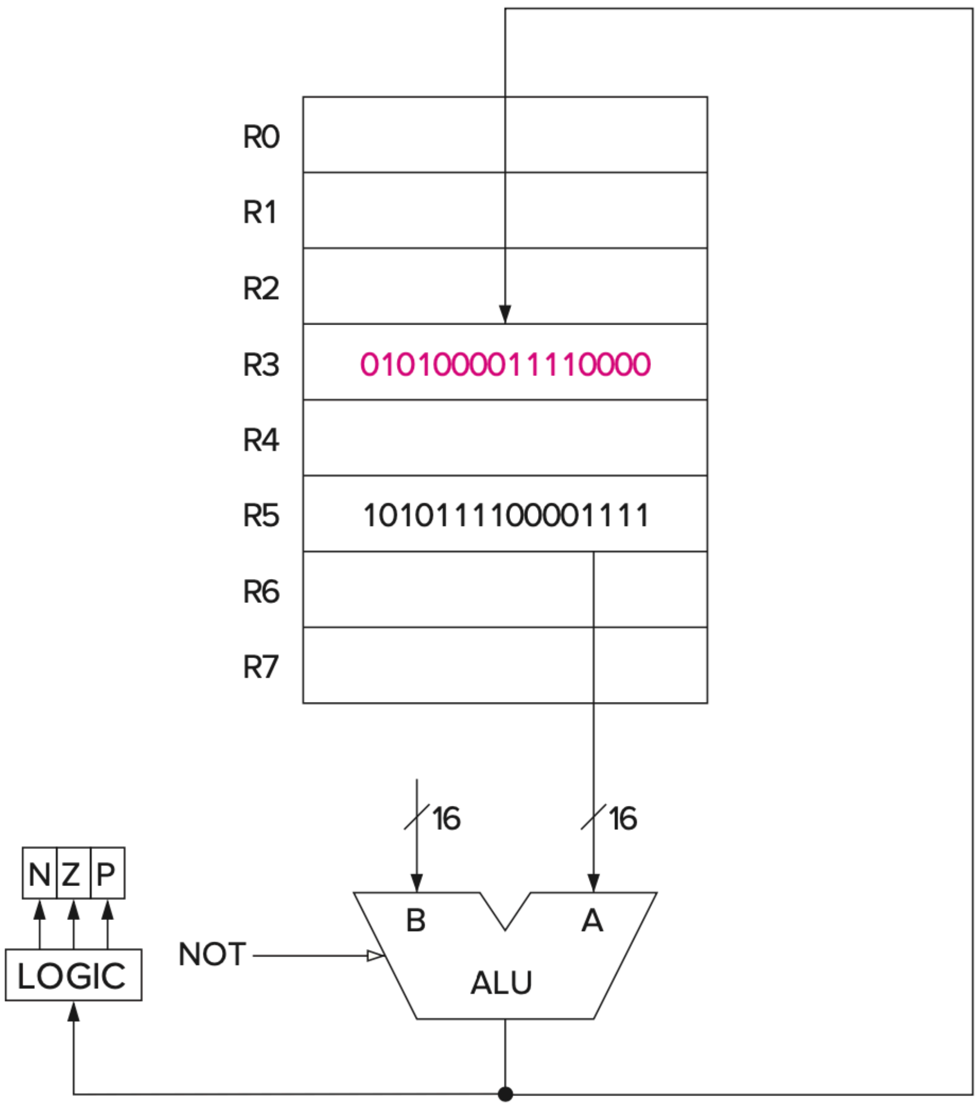
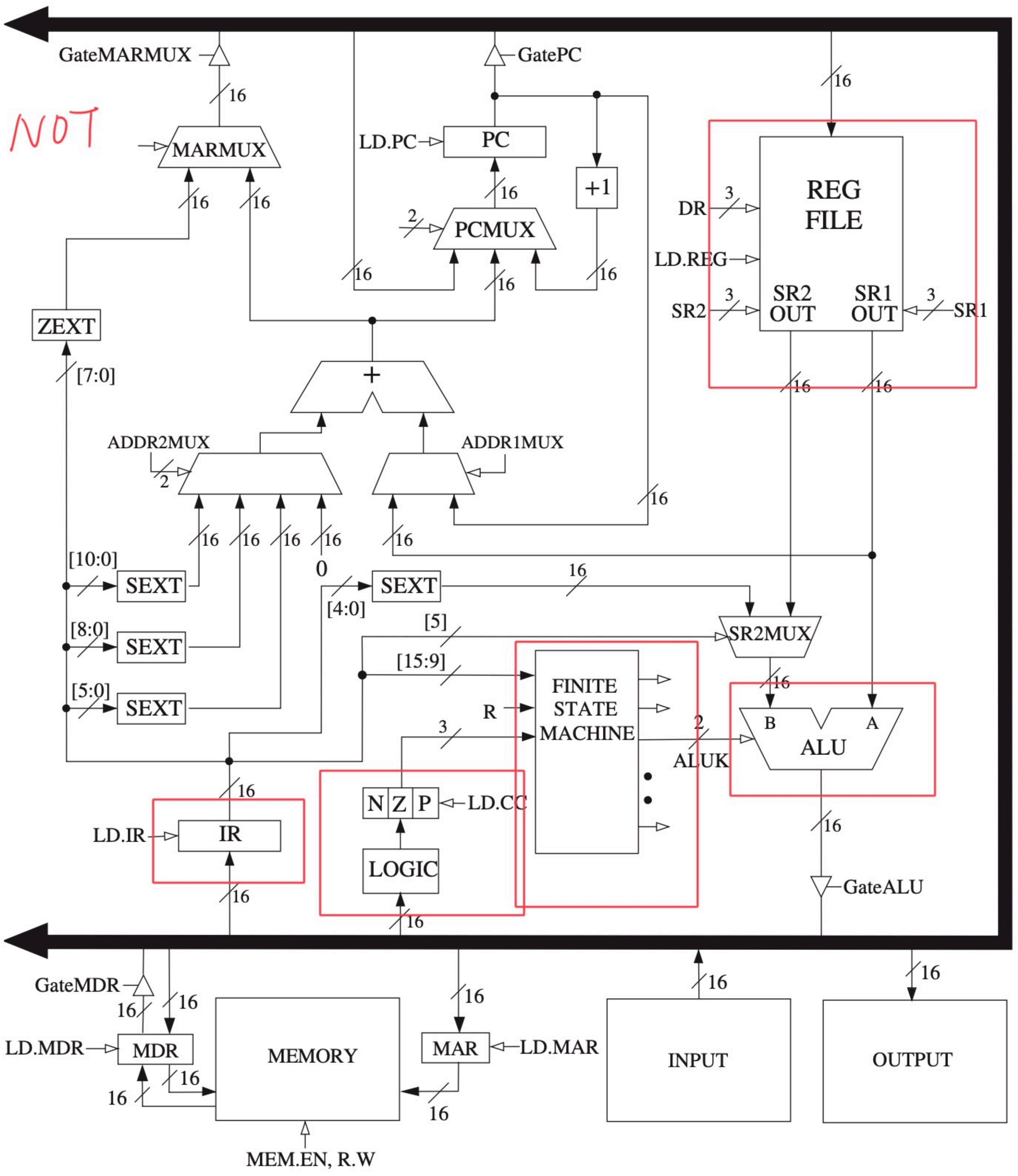


Figure 5.4 **Data path relevant to the execution of NOT R3, R5.**

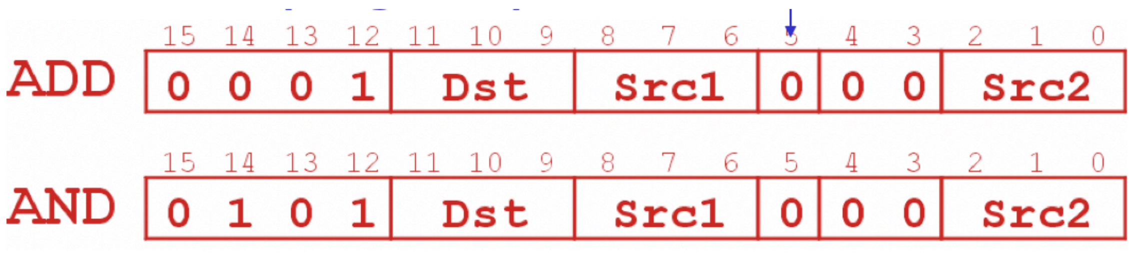
> ALU: NOT,AND,ADD and PASS

data path



5.2.2 AND/ADD with Registers

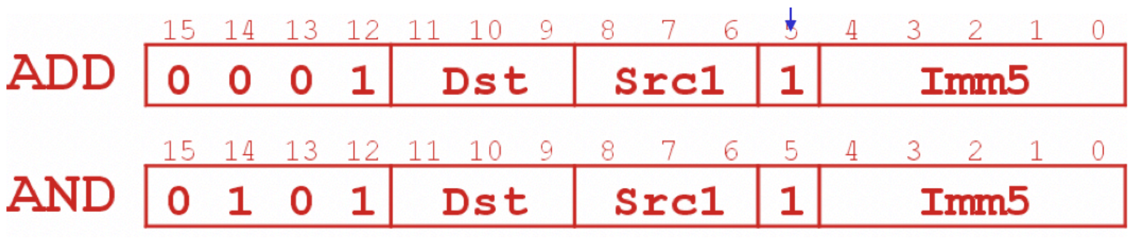
machine code



	15 14 13 12	11 10 9	8 7 6	5	4 3	2 1 0
ADD	0 0 0 1	Dst	Src1	0	0 0	Src2
AND	0 1 0 1	Dst	Src1	0	0 0	Src2

5.2.3 AND/ADD with imm

machine code



	15 14 13 12	11 10 9	8 7 6	5	4 3 2 1 0
ADD	0 0 0 1	Dst	Src1	1	Imm5
AND	0 1 0 1	Dst	Src1	1	Imm5

dataflow

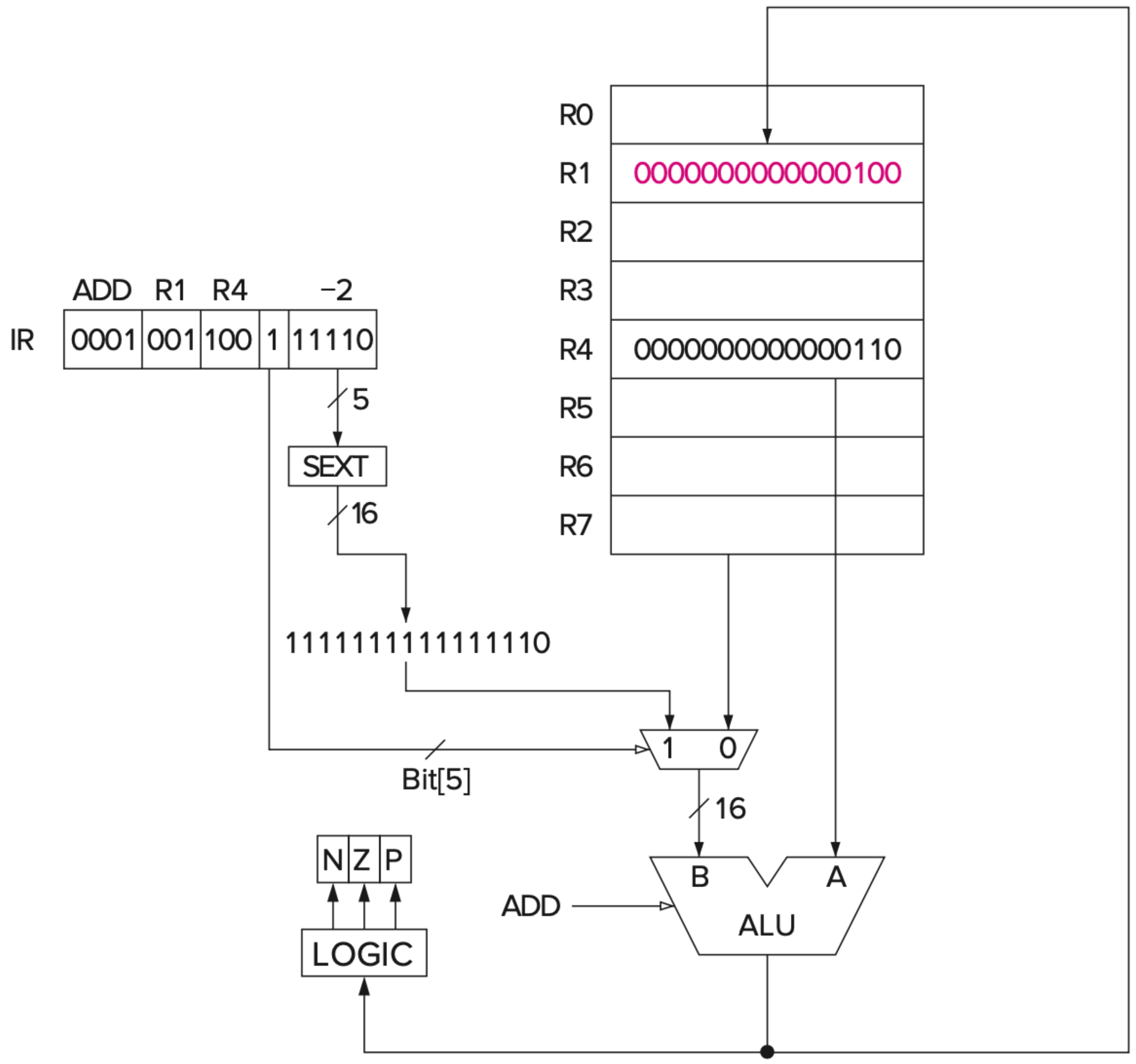
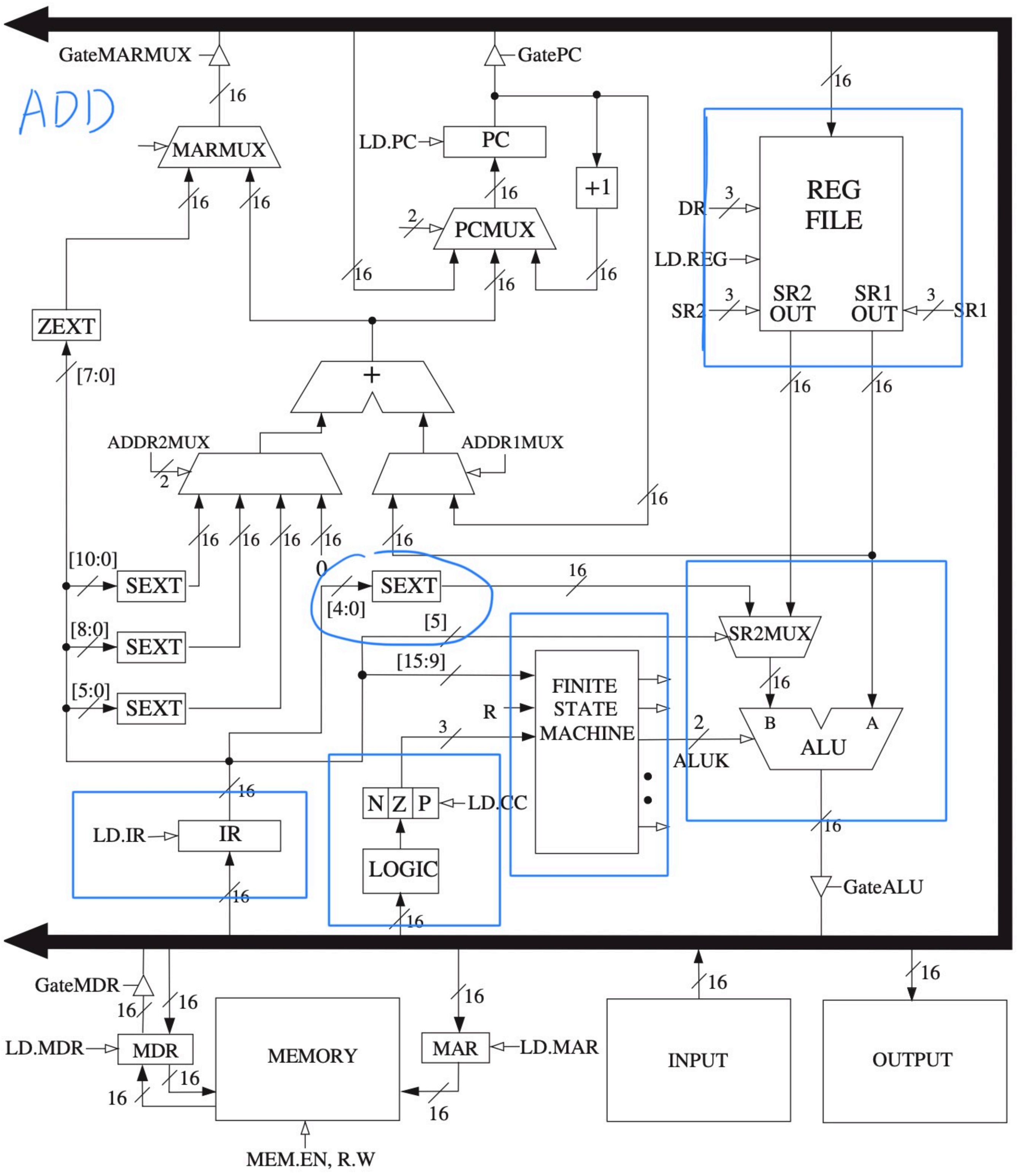


Figure 5.5 **Data path relevant to the execution of ADD R1, R4, #−2.**

data path



5.2.4 Using Examples

Subtract

```
// R1 <- R0 - R1
1001 001 001 111111   // NOT  R1, R1
0001 001 001 1 00001  // ADD  R1, R1, #1
0001 001 000 000 001  // ADD  R1, R0, R1
```

OR

```
/* R1 = R1 OR R0 */
1001 001 001 111111   //NOT R1, R1
1001 000 000 111111   //NOT R0, R0
0101 001 001 000 000  //AND R1, R1, R0
1001 001 001 111111   //NOT R1, R1
```

Bitwise Left Shift

Register Copy

Initialize 0

5.3 Data Movement Instructions

5.3.1 LD/ST - PC Relative

The Calculation of offset

9offset : [-256, +255] away from PC

A address xYYYY

B address xZZZZ

A go to B:

PC = xYYYY + 1

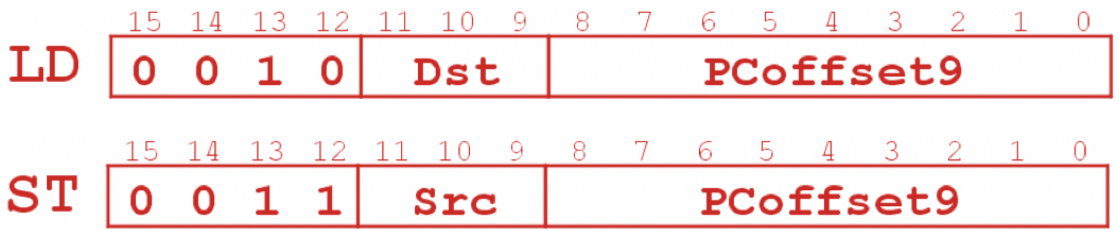
offset = xZZZZ - xYYYY - 1 -> 保留9bits

两个地址直接相减，再减1，保留9位

> *Remember that PC is incremented as part of the FETCH phase;*
>
> *This is done before the EVALUATE ADDRESS stage.*

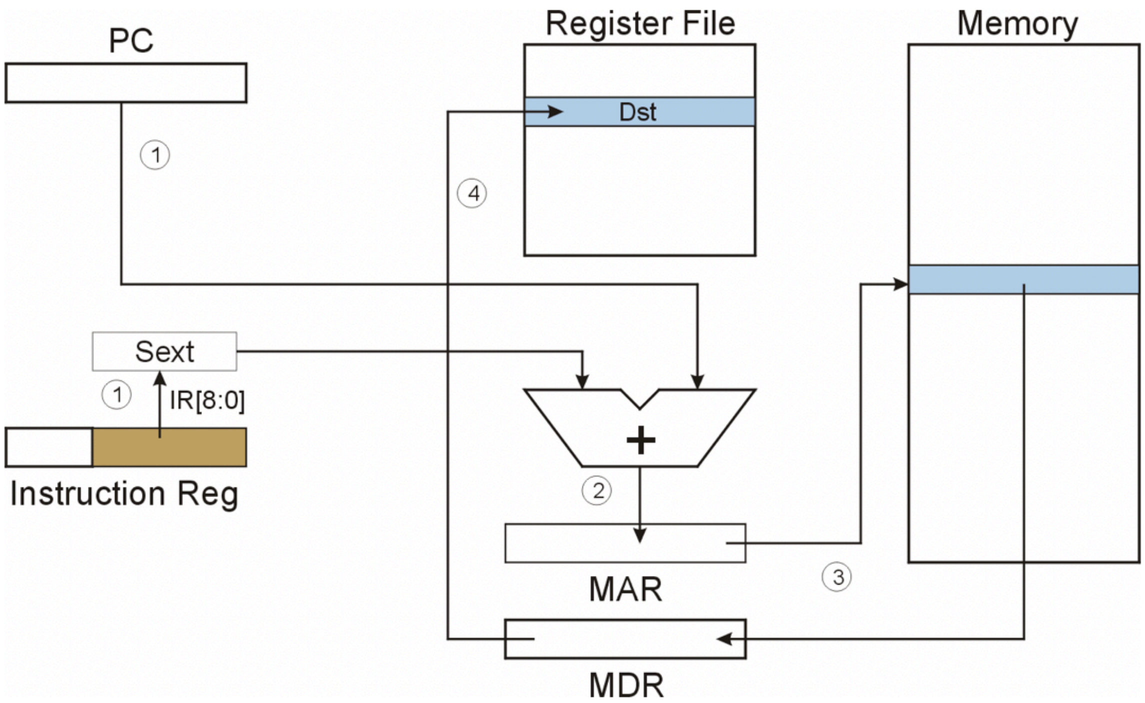
Note: offset whether 9bits or 6bits, is **signed integer**, so pay attention to overflow

machine code

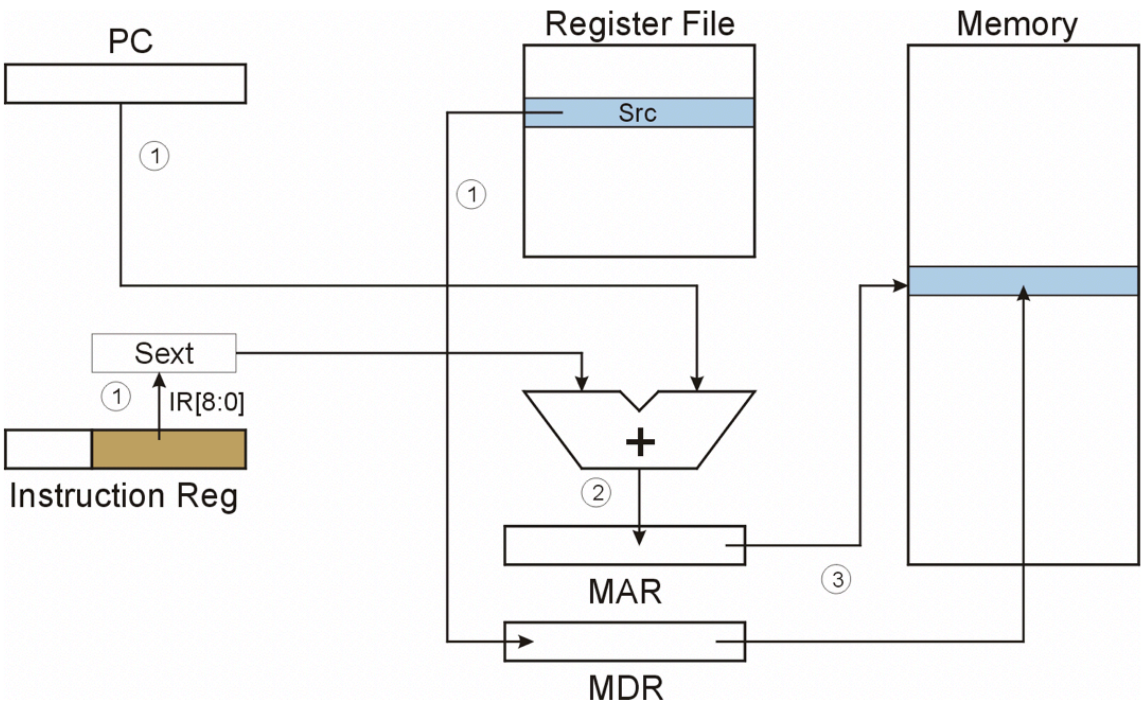


dataflow

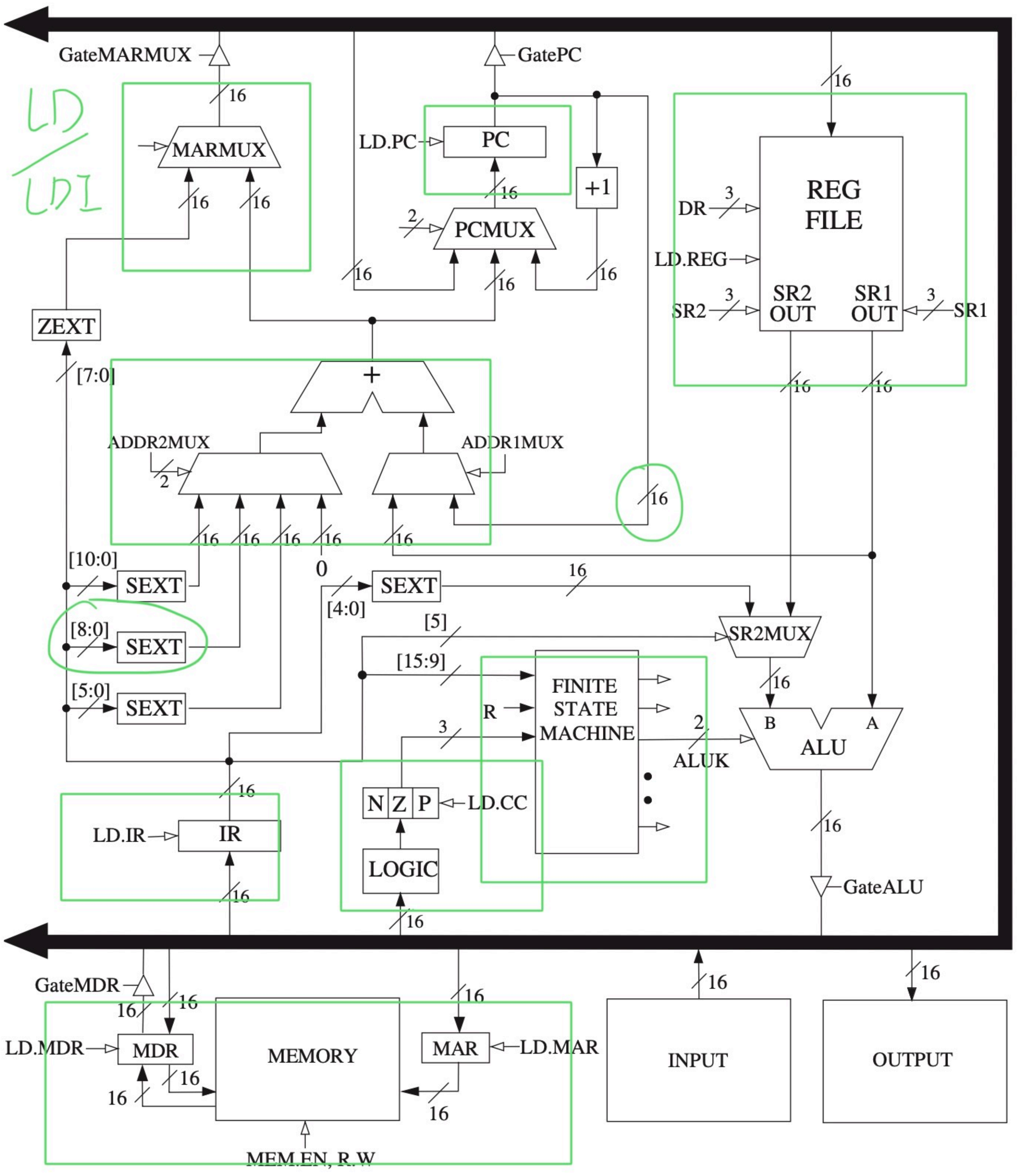
LD


PC
Register File
Memory
Dst
①
④
Sext
①
IR[8:0]
+
②
Instruction Reg
MAR
③
MDR

ST


PC
Register File
Memory
Src
①
①
Sext
①
IR[8:0]
+
②
Instruction Reg
MAR
③
MDR

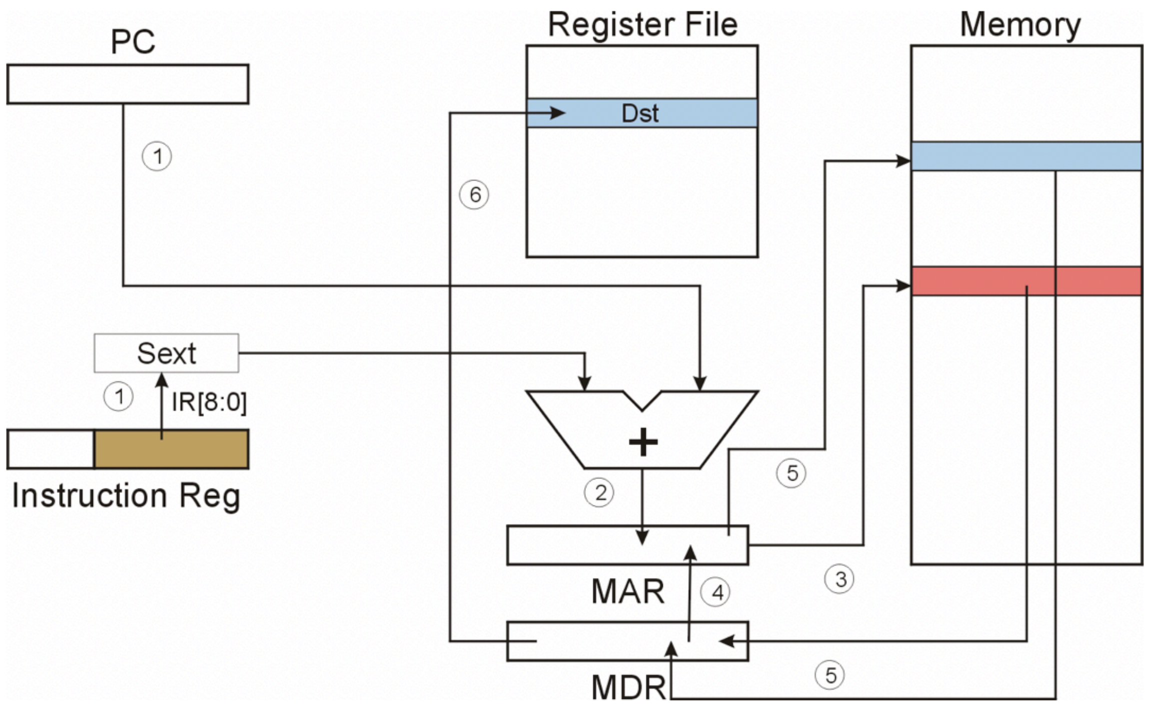
data path



5.3.2 LDI/STI - Indirect

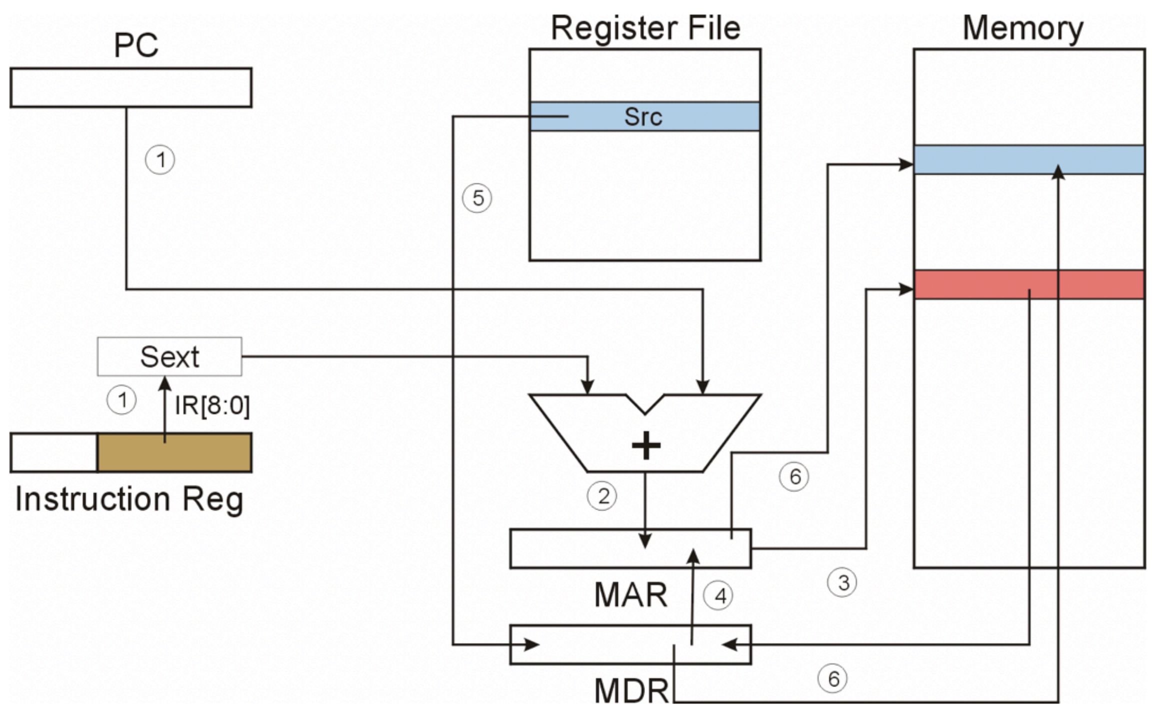
LDI (Indirect)

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



STI (Indirect)

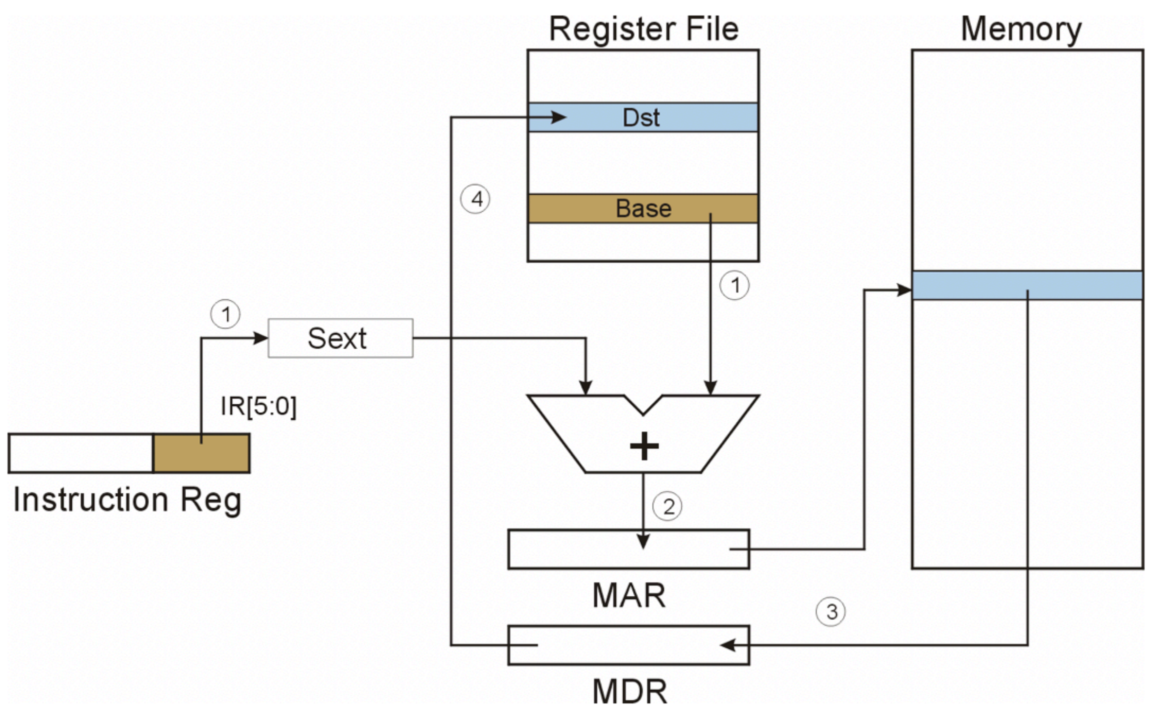
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STI	1	0	1	1	Src			PCoffset9								

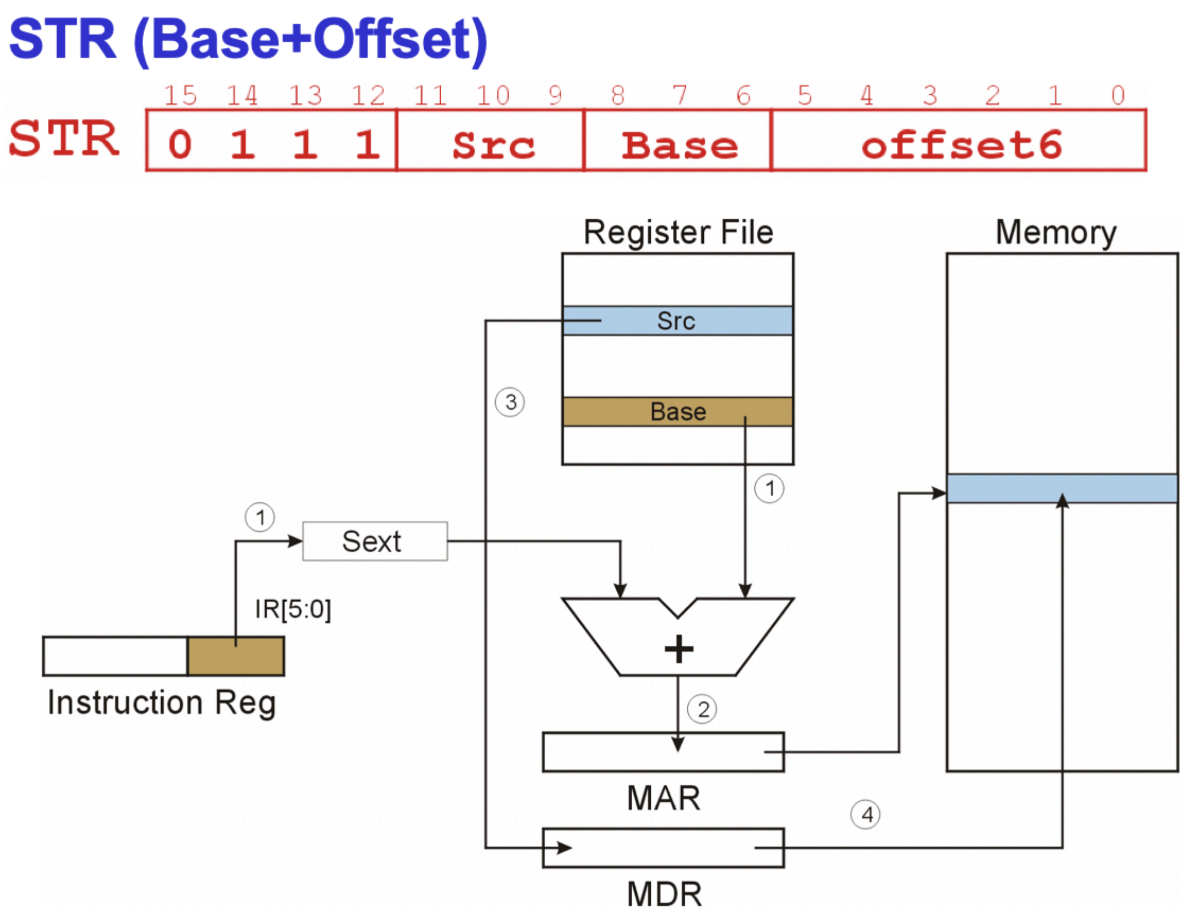


5.3.3 LDR/STR - Base+Offset

LDR (Base+Offset)

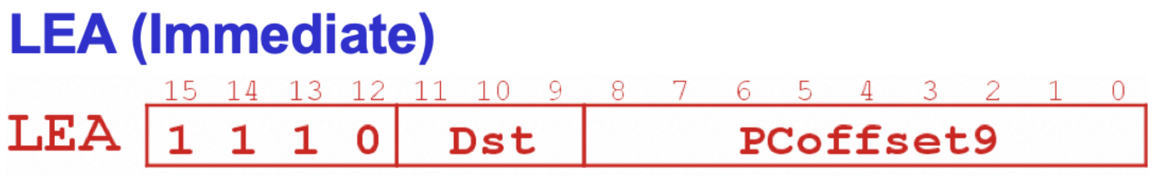
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDR	0	1	1	0	Dst			Base			offset6					

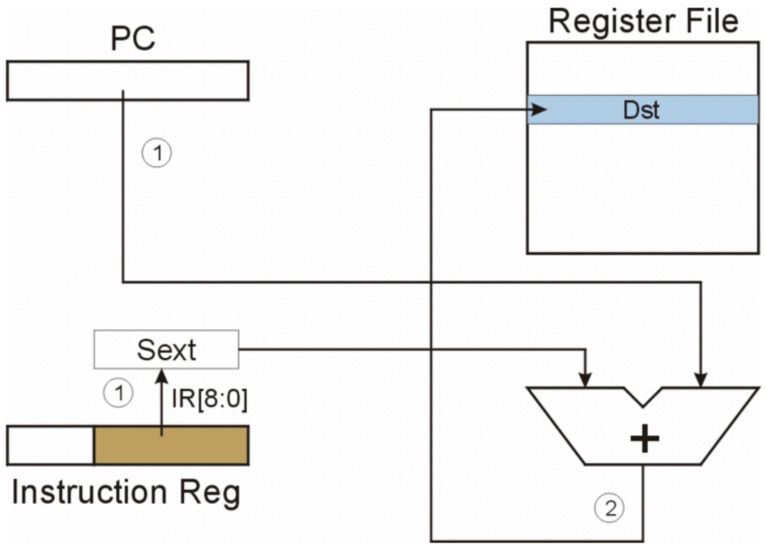




5.3.4 LEA - Immediate

> LEA, stands for *Load effective address*
>
> don't access memory
>
> **address** stored into the register





5.4 Control Instructions

5.4.1 Conditional Branches

BR	N	Z	P	PC+offset	Condition	Take Branch
[15:12]	[11]	[10]	[9]	[8:0]		
0000	1	0	1		Check N,P CC	N or P is set(i.e. Not 0)
	1	1	1		Check N,Z,P	Always(Unconditional)
	0	0	0			Never

If bit [9] is 1, condition code P is examined. If any of bits [11:9] are 0, the associated condition codes are not examined. If any of the condition codes that are examined are set (i.e., equal to 1), then the PC is loaded with the address obtained in the EVALUATE ADDRESS phase. If none of the condition codes that are examined are set, the incremented PC is left unchanged, and the next sequential instruction will be fetched at the start of the next instruction cycle.

5.4.2 Two Methods of Loop Control

5.4.3 JUMP

5.4.4 TRAP

TRAP(1 1 1 1)	**0 0 0 0**	**trap-vector**