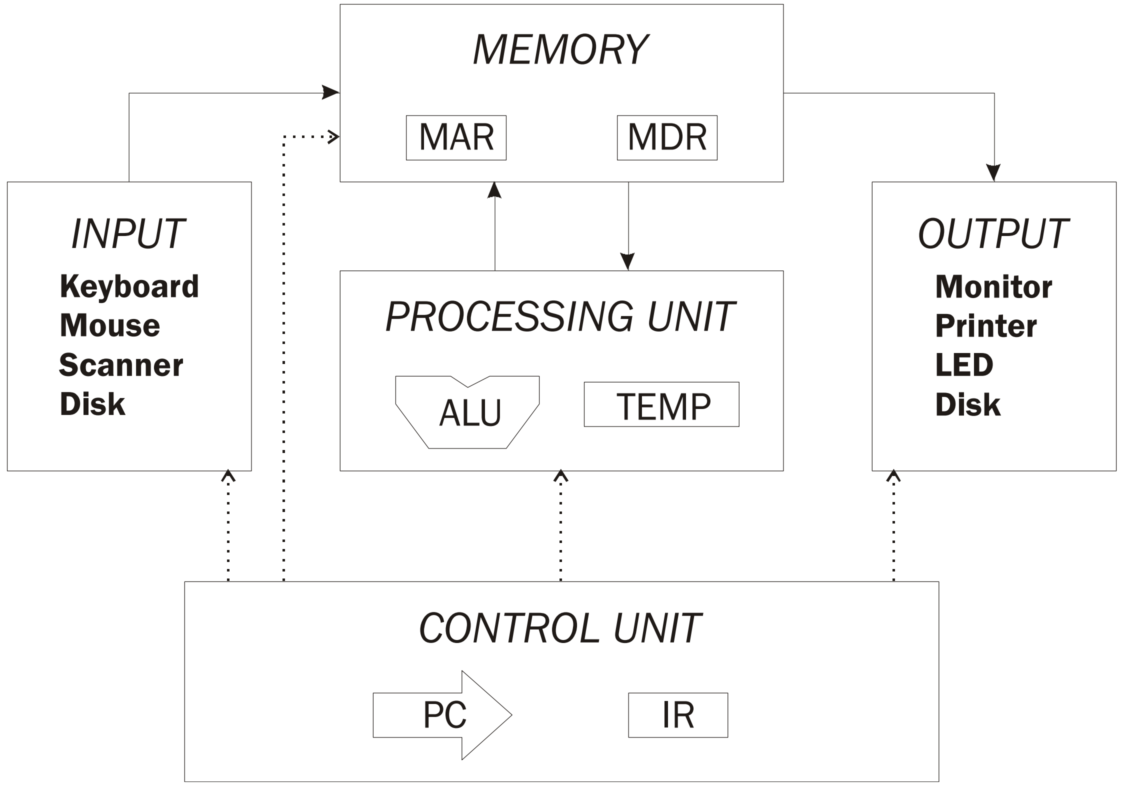
310-2202 โครงสร้างของระบบคอมพิวเตอร์ (Computer Organization)

Topic 5: The LC-3 Instructions: Data Movement Instructions

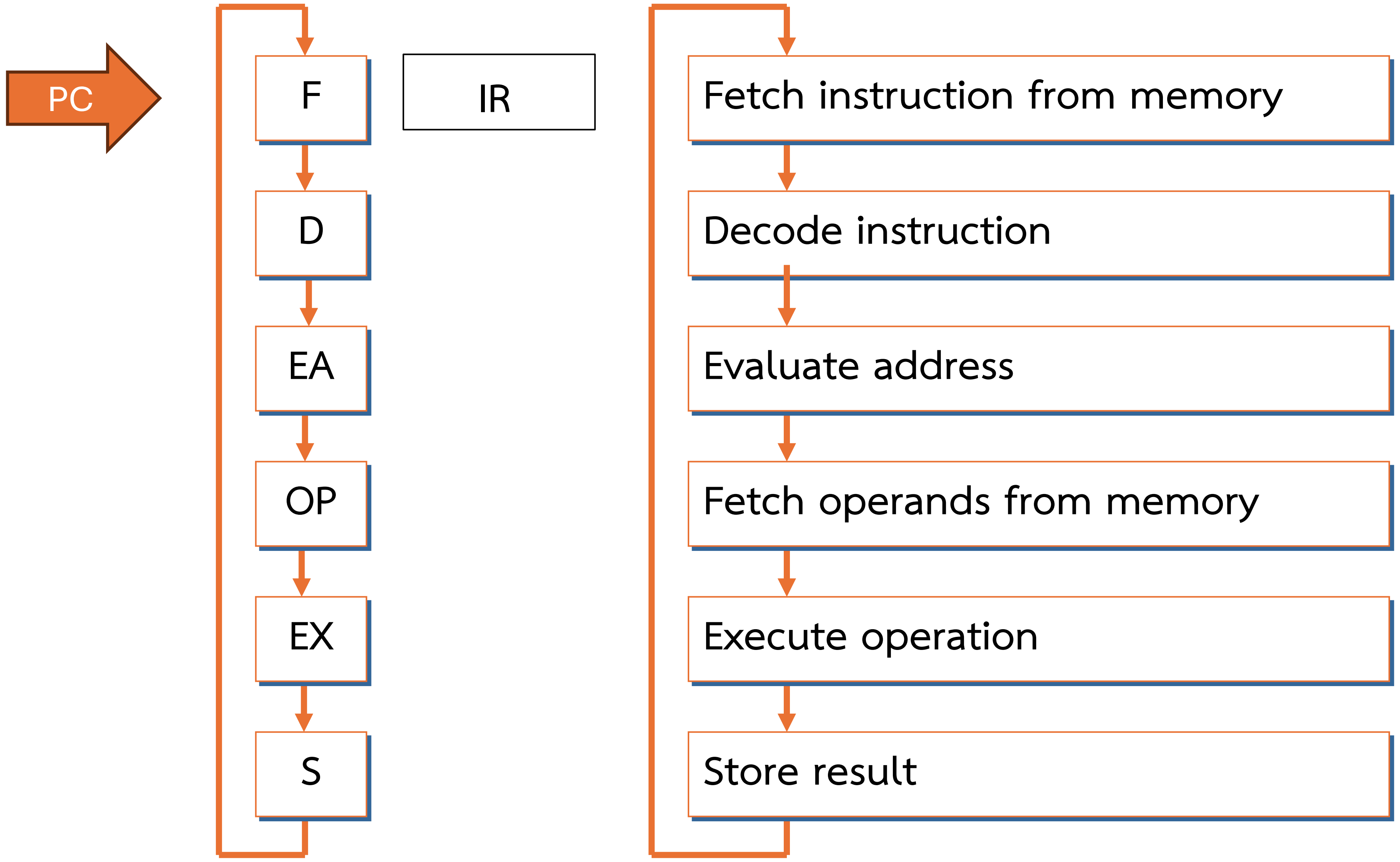
Damrongrit Setsirichok

Topic

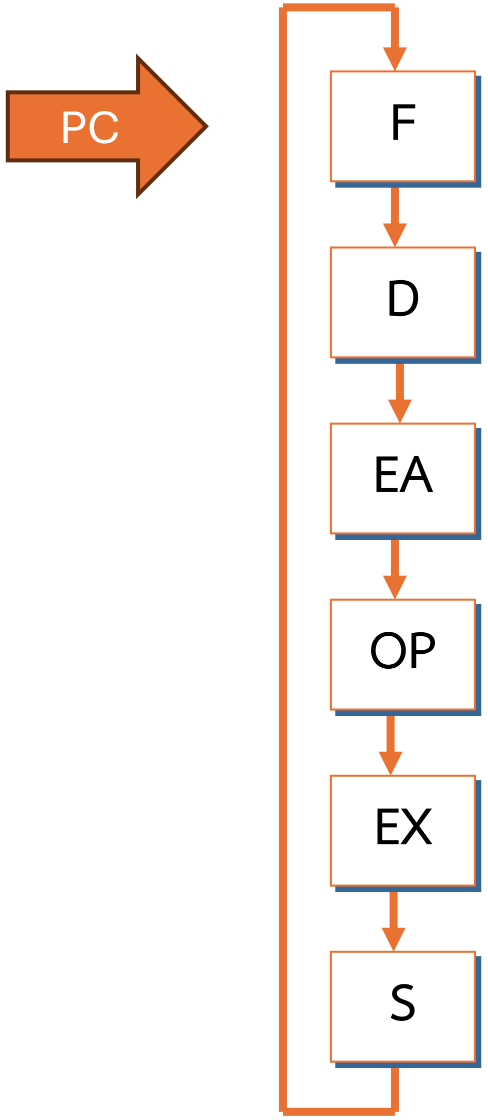
- LC-3 Data Movement Instructions
  - LC-3 PC-Relative Load/Store
  - LC-3 Indirect, Base+offset Load/Store



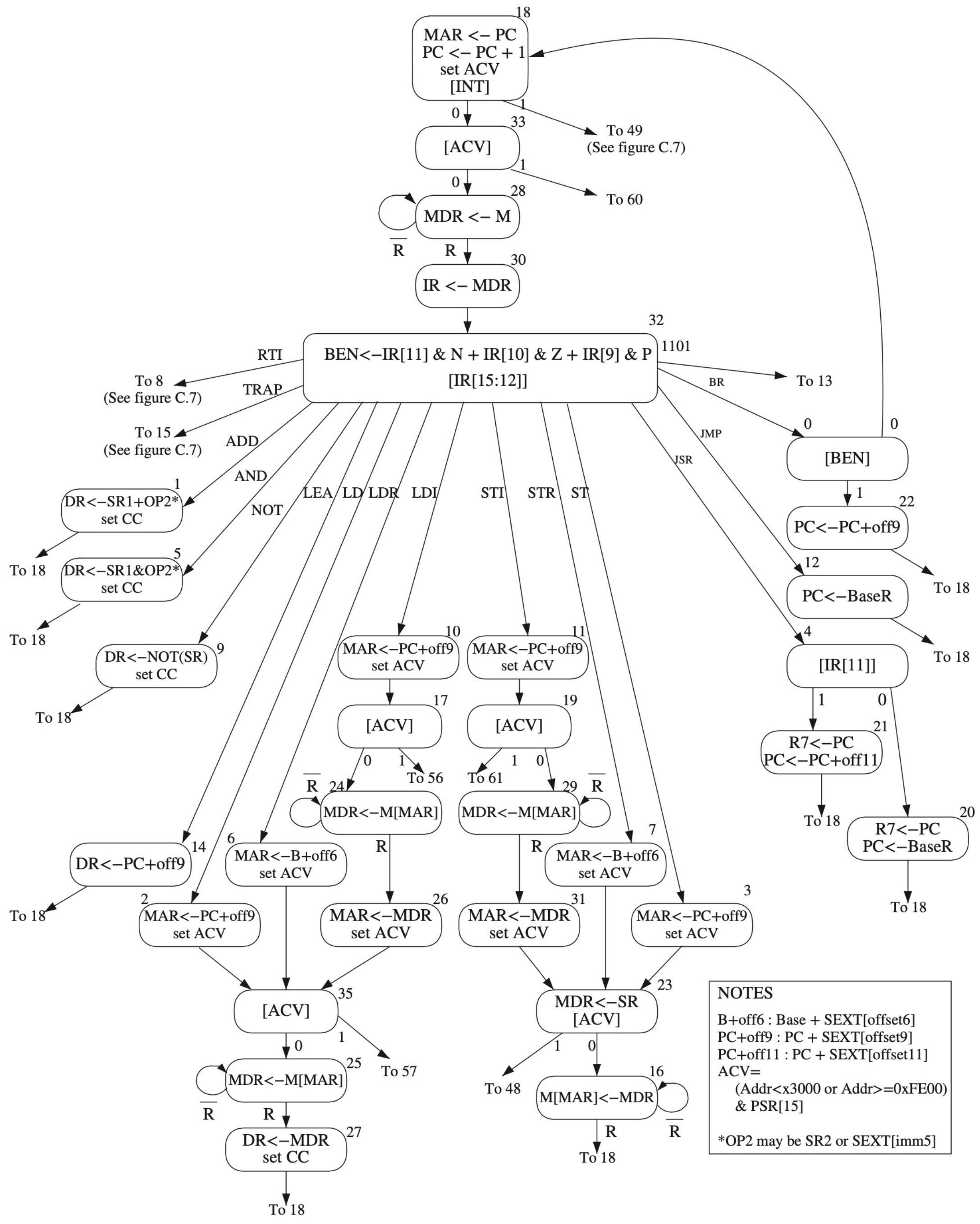
Instruction Processing: State Transition



Instruction Processing: Finite State Automata



702 appendix C The Microarchitecture of the LC-3

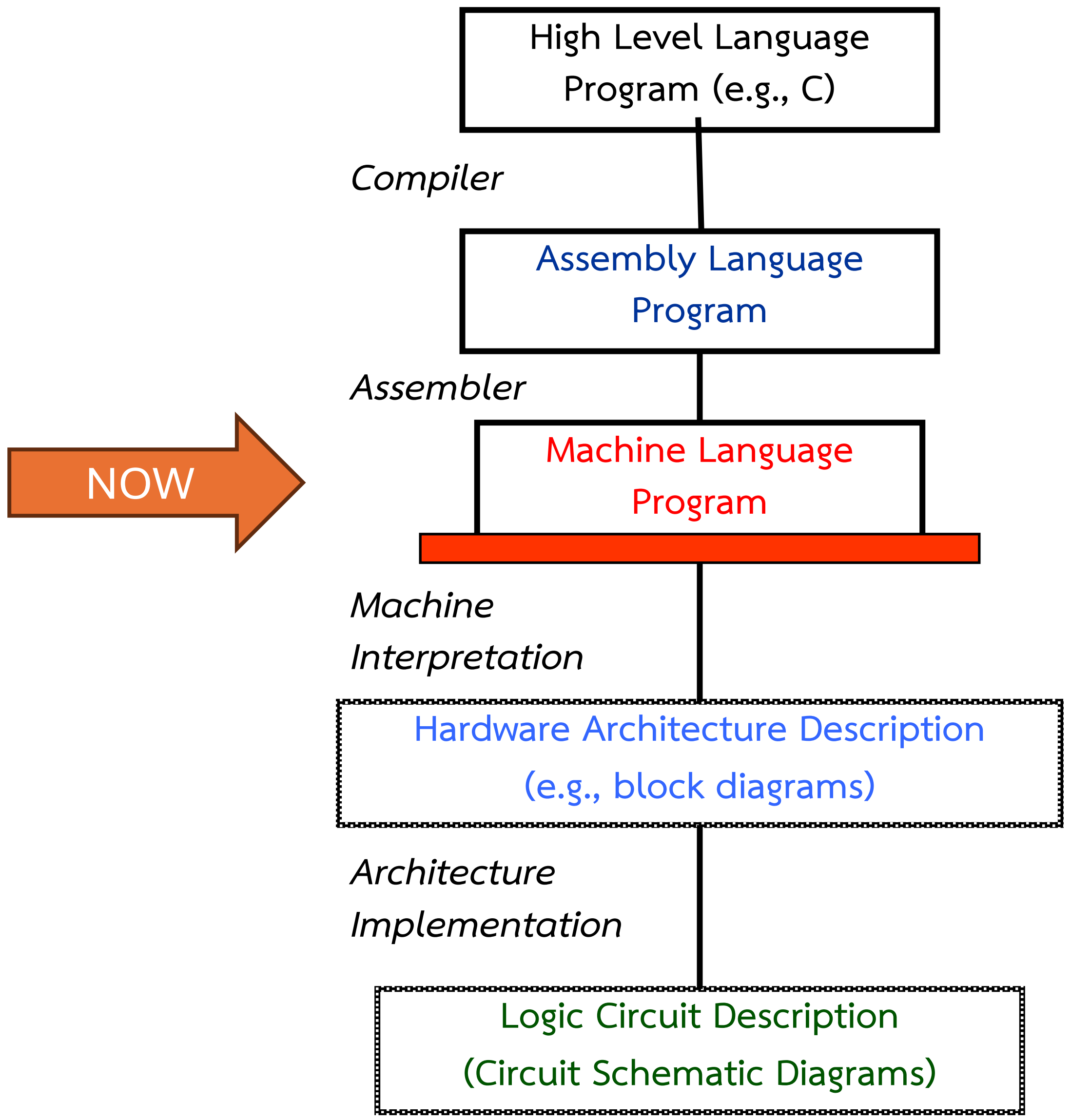


NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT[offset9]
PC+off11 : PC + SEXT[offset11]
ACV=
(Addr<x3000 or Addr>=0xFE00) & PSR[15]

*OP2 may be SR2 or SEXT[imm5]

Figure C.2 A state machine for the LC-3.

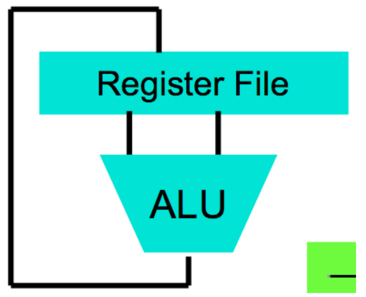
How do we get the electrons to do the work?

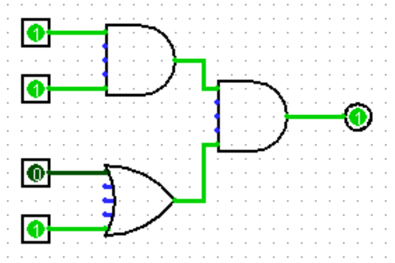

High Level Language Program (e.g., C)
Compiler
Assembly Language Program
Assembler
NOW
Machine Language Program
Machine Interpretation
Hardware Architecture Description (e.g., block diagrams)
Architecture Implementation
Logic Circuit Description (Circuit Schematic Diagrams)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw  $t0, 0(a0)
lw  $t1, 4(a0)
sw  $t1, 0(a0)
sw  $t0, 4(a0)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



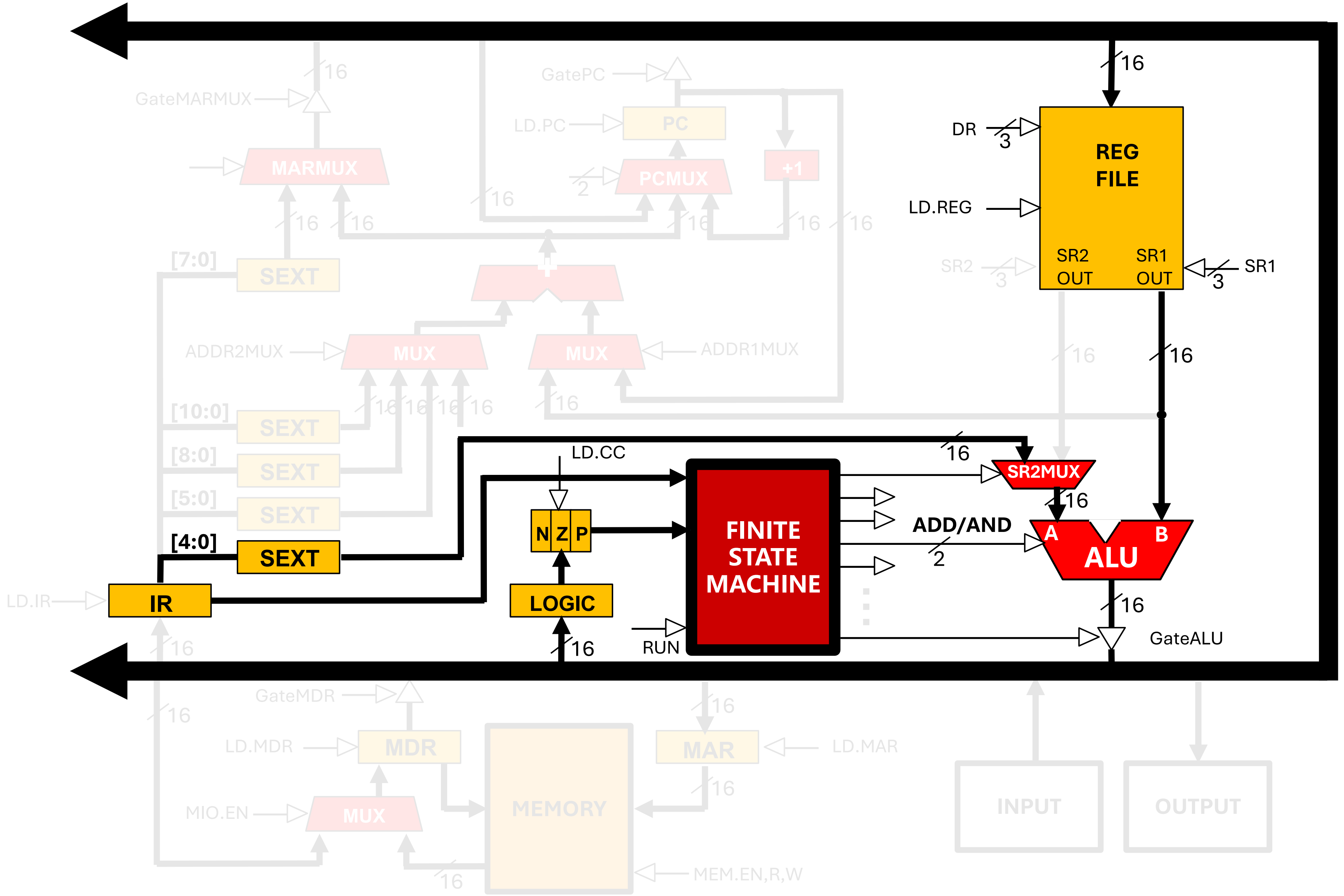


LC-3 Data Movement Instructions

LC-3 PC-Relative Load/Store

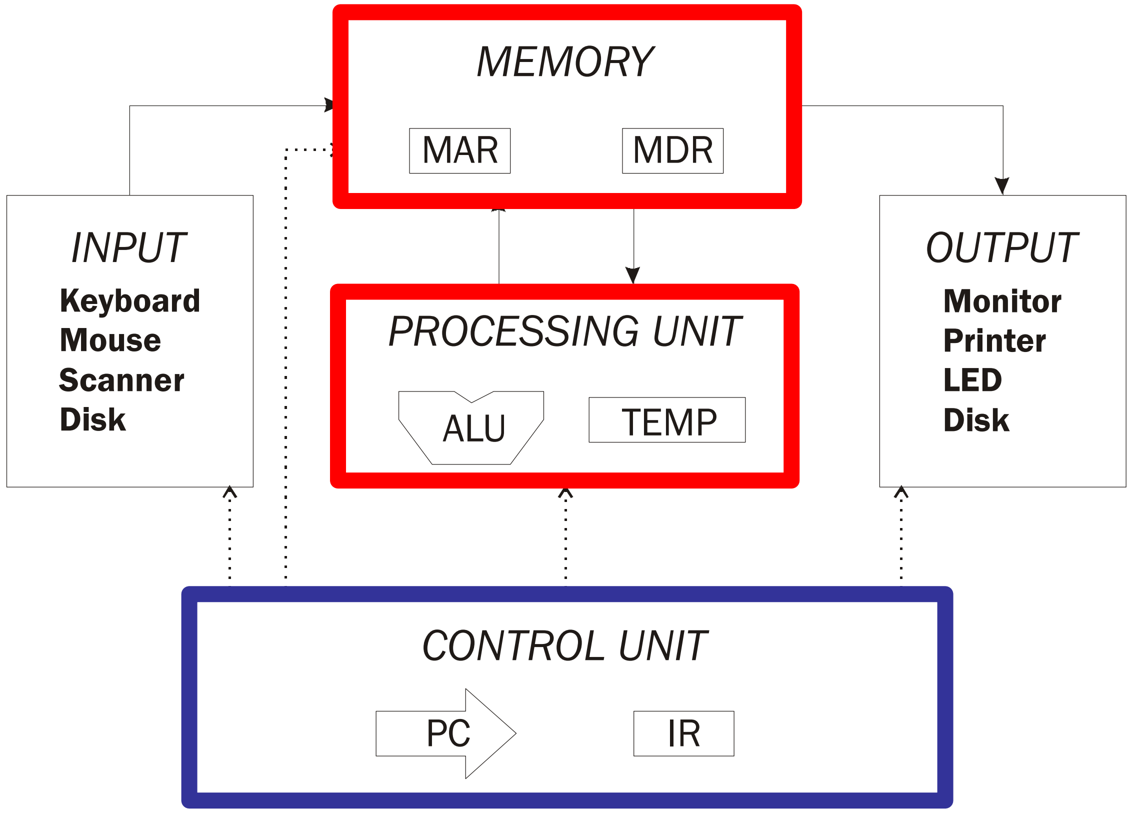
LC-3 Indirect, Base+offset Load/Store

LC-3 Data Path After Operate Instruction

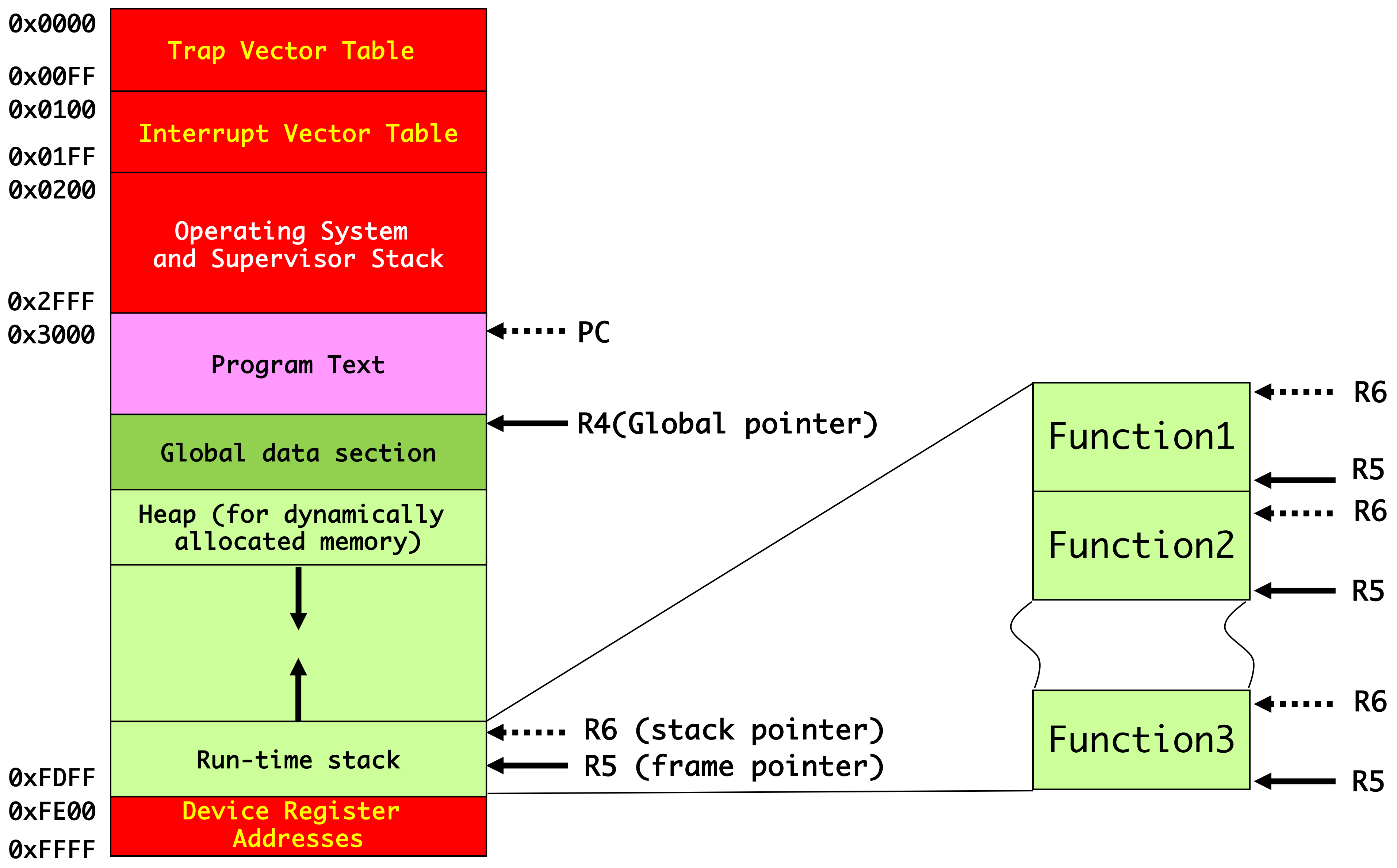


LC-3 PC-Relative Load/Store

- **Load** data from memory to registers
- **Store** data from registers to memory



LC-3 Overview: Memory Map



LC-3 Data Movement Instructions

Data Movement Instructions

Load

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LD	0	0	1	0	DR			PCoffset9								
LDR	0	1	1	0	DR			BaseR			PCoffset6					
LDI	1	0	1	0	DR			PCoffset9								
LEA	1	1	1	0	DR			PCoffset9								

Store

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST	0	0	1	1	SR			PCoffset9								
STR	0	1	1	1	SR			BaseR			PCoffset6					
STI	1	0	1	1	SR			PCoffset9								

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 Addressing Mode

- Specify address directly in the instruction
  - After subtracting 4 bits for opcode and 3 bits for register, we have 9 bits available for address → **Address is 16 bits**
  - Solution → Use the 9 bits as a *signed offset* from the current PC

9 bits: **$-256 \le \text{offset} \le +255$**

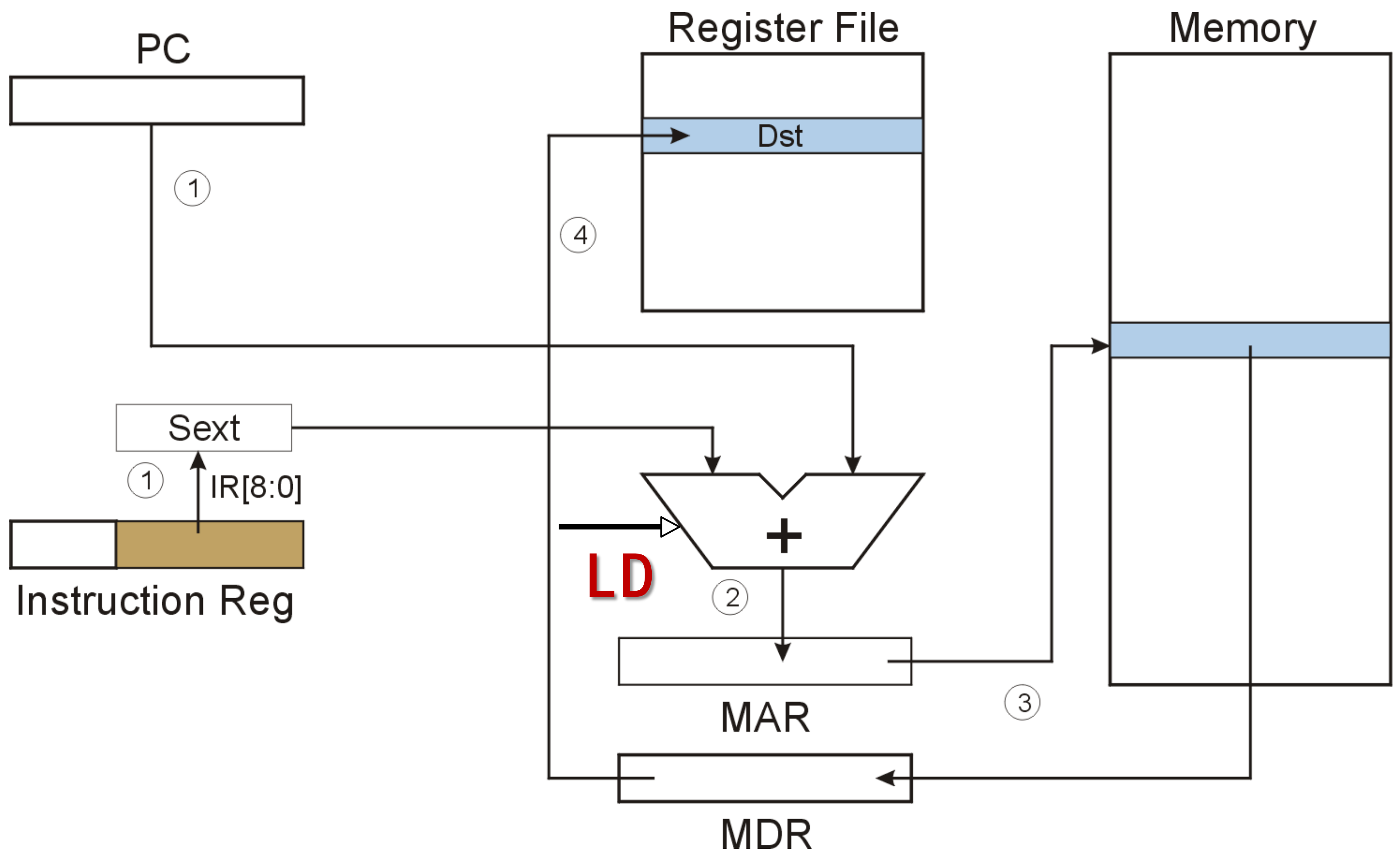
Can form any address X, such that: **$(PC - 256) \le X \le (PC + 256)$**

*** PC is incremented as part of the FETCH phase;

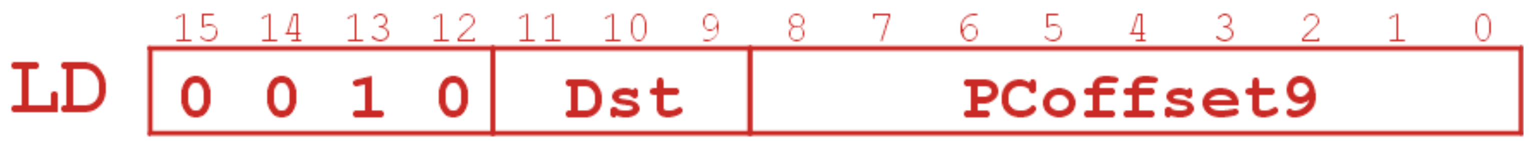
This is done before the EVALUATE ADDRESS stage ***

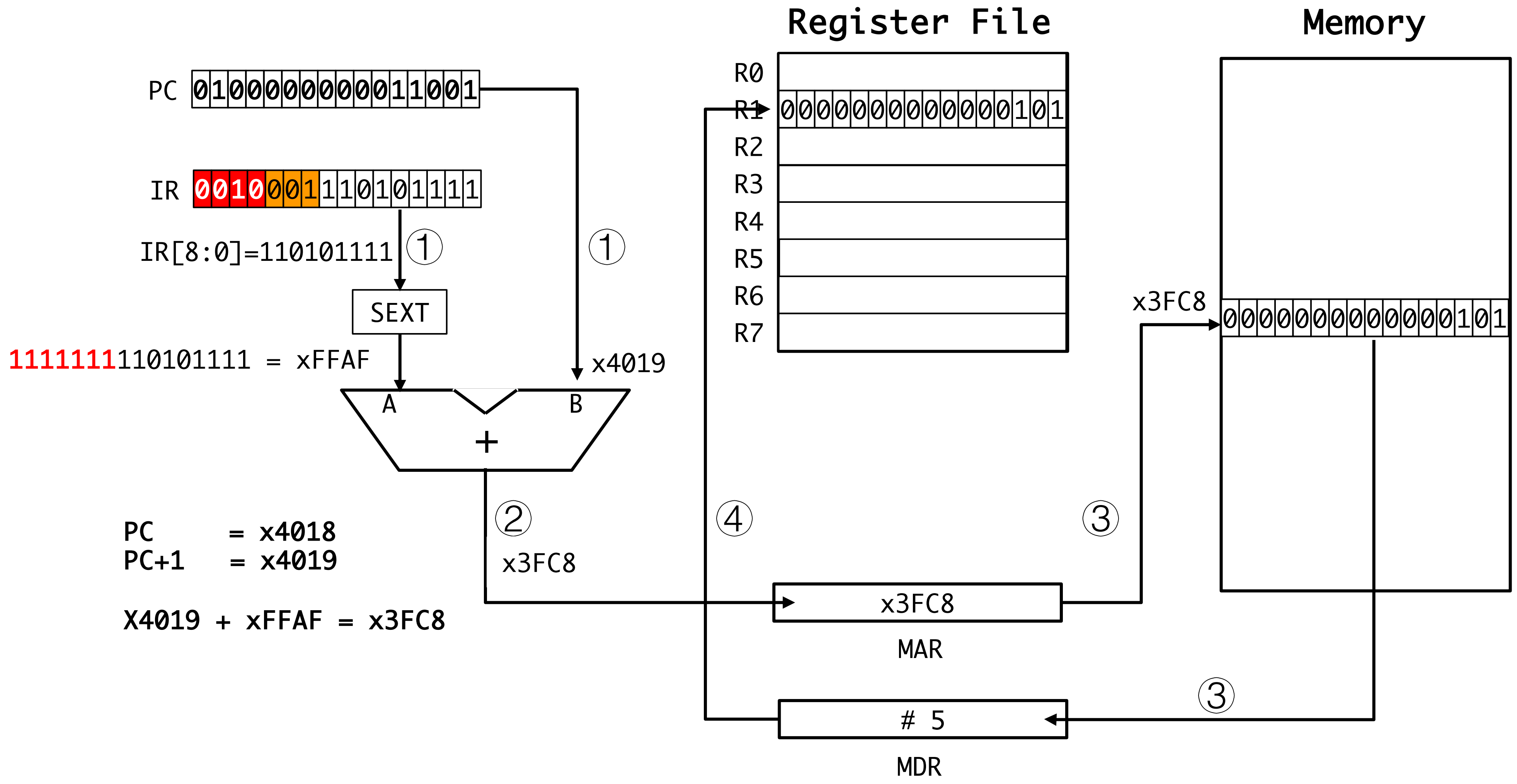
LD (PC-Relative)   LD DR, PCoffset9

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



LD (PC-Relative) : LD R1, x1AF

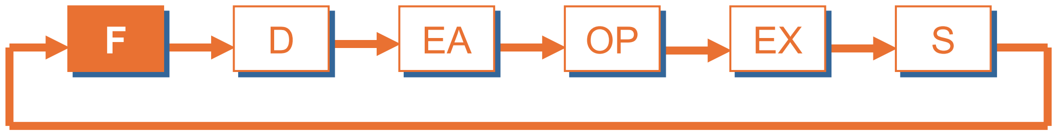


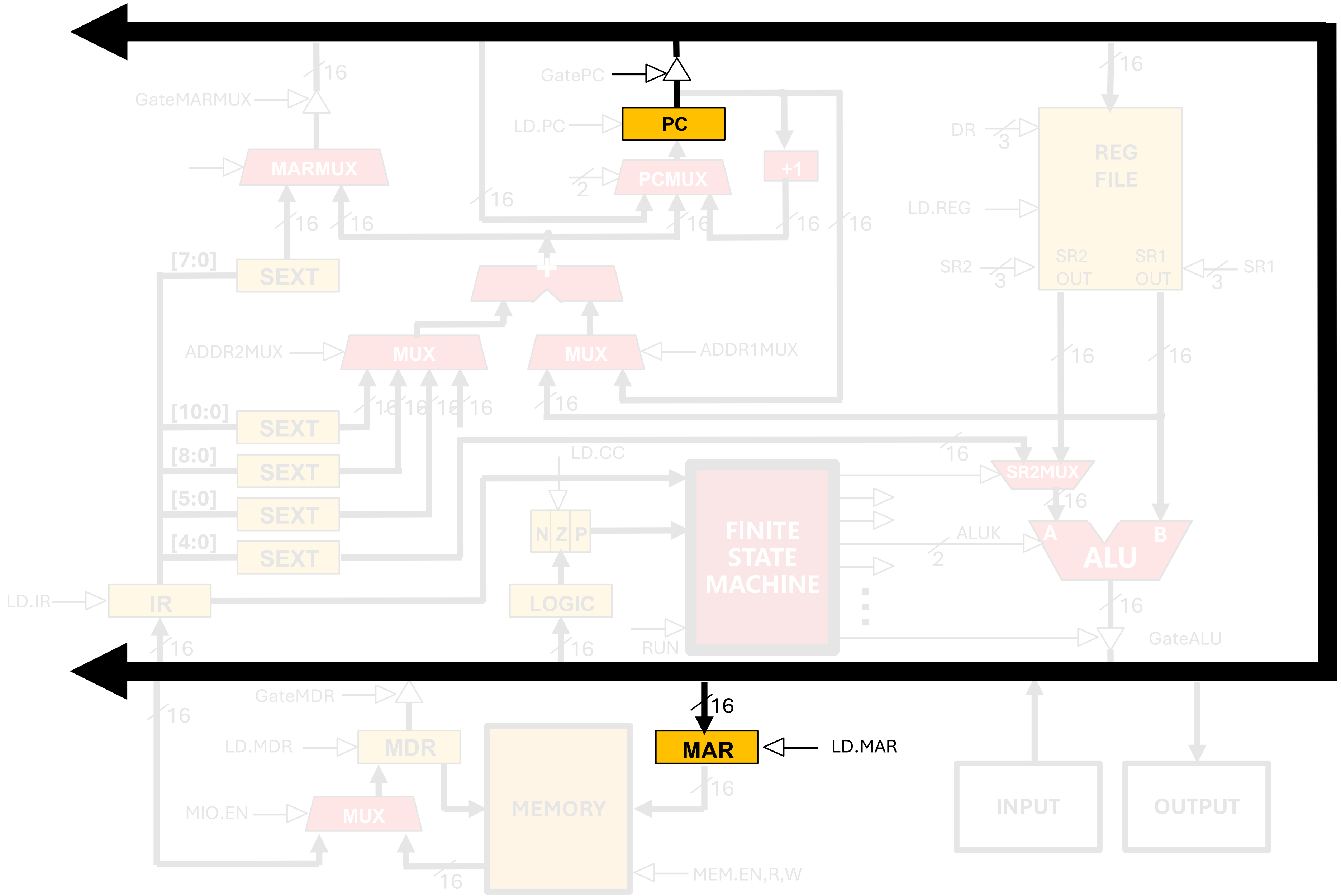


PC = x4018
PC+1 = x4019

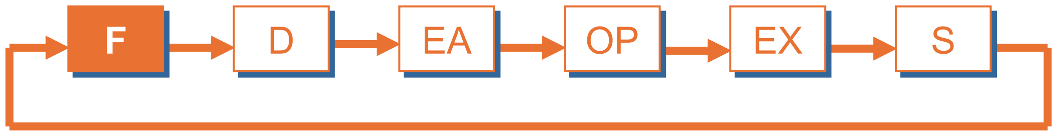
X4019 + xFFAF = x3FC8

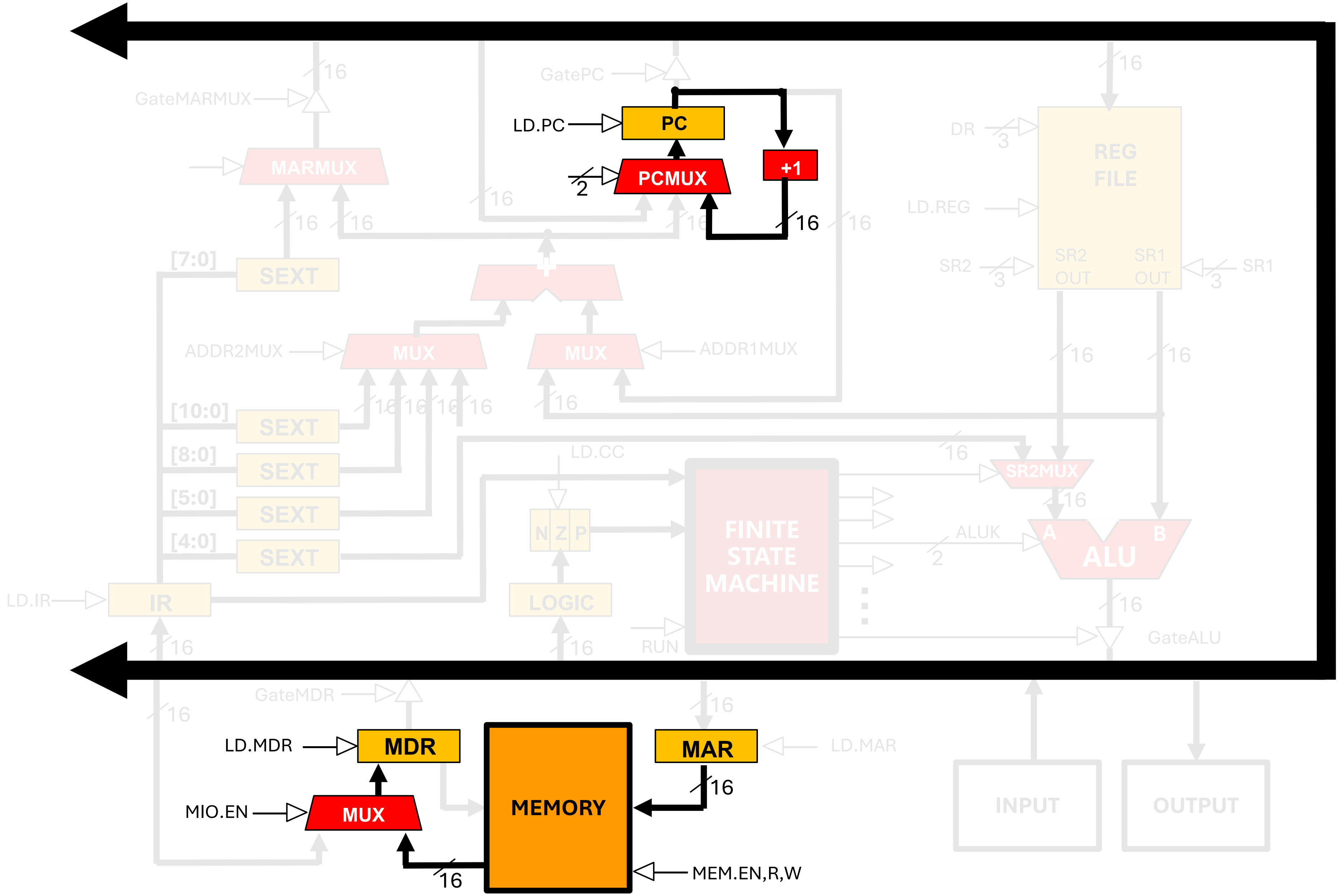
LD (PC-Relative)



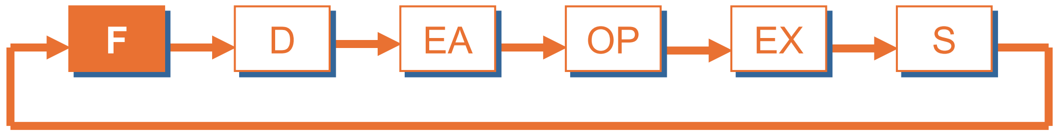


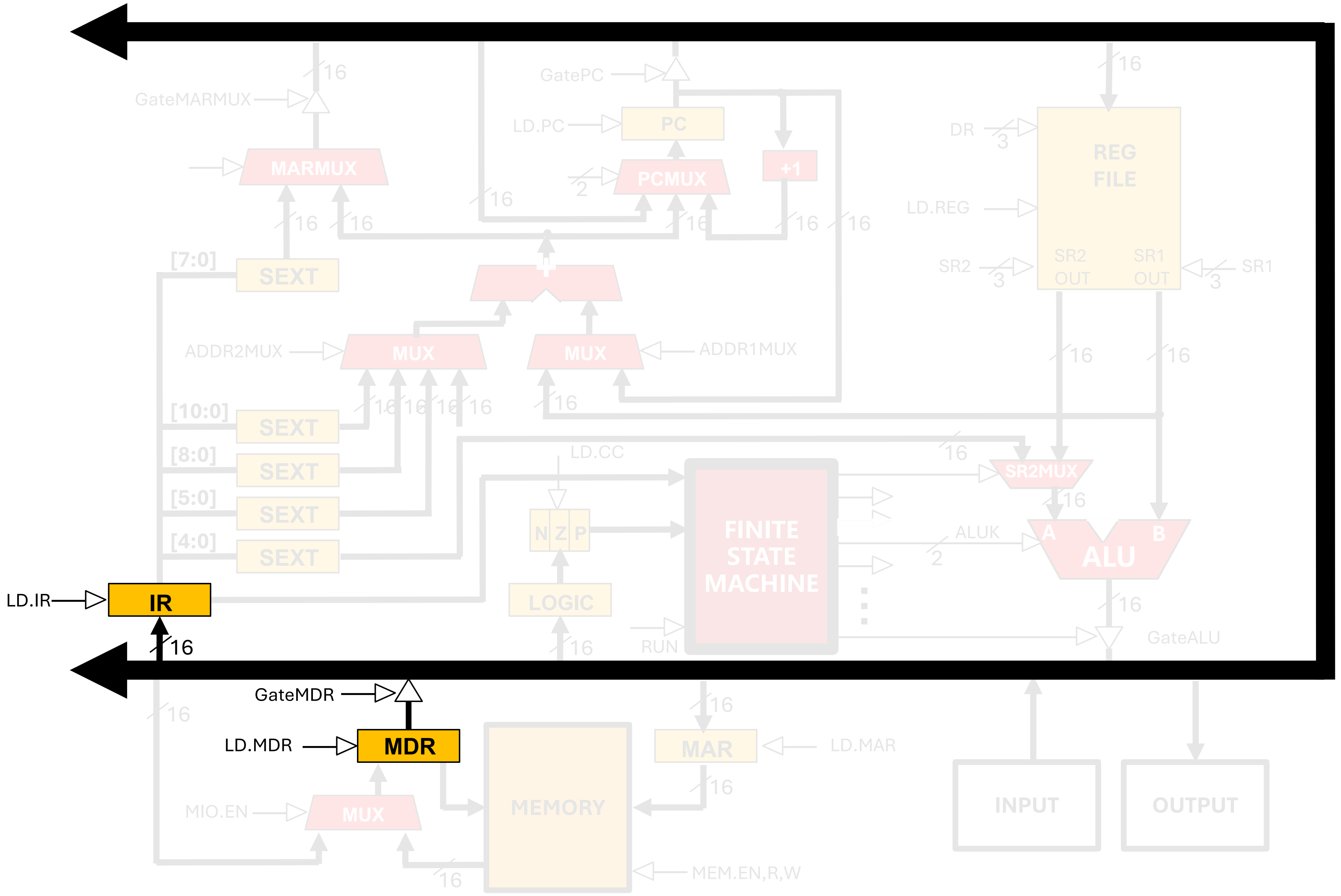
LD (PC-Relative)



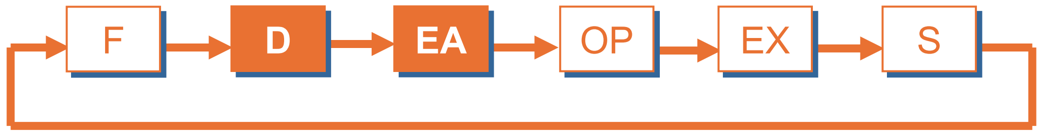


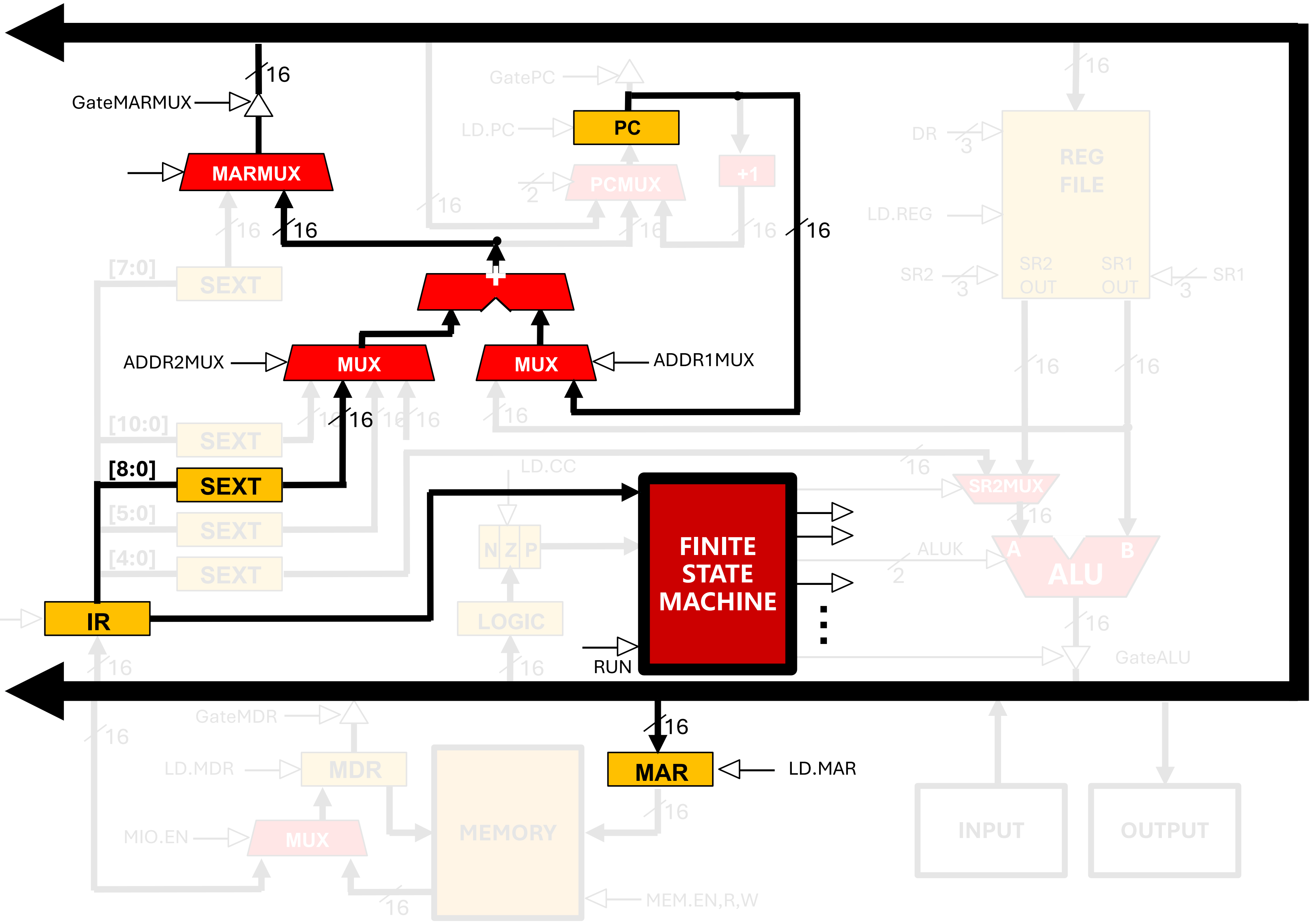
LD (PC-Relative)



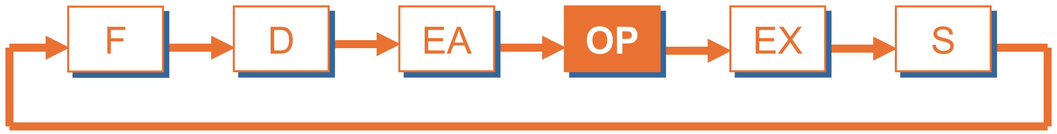


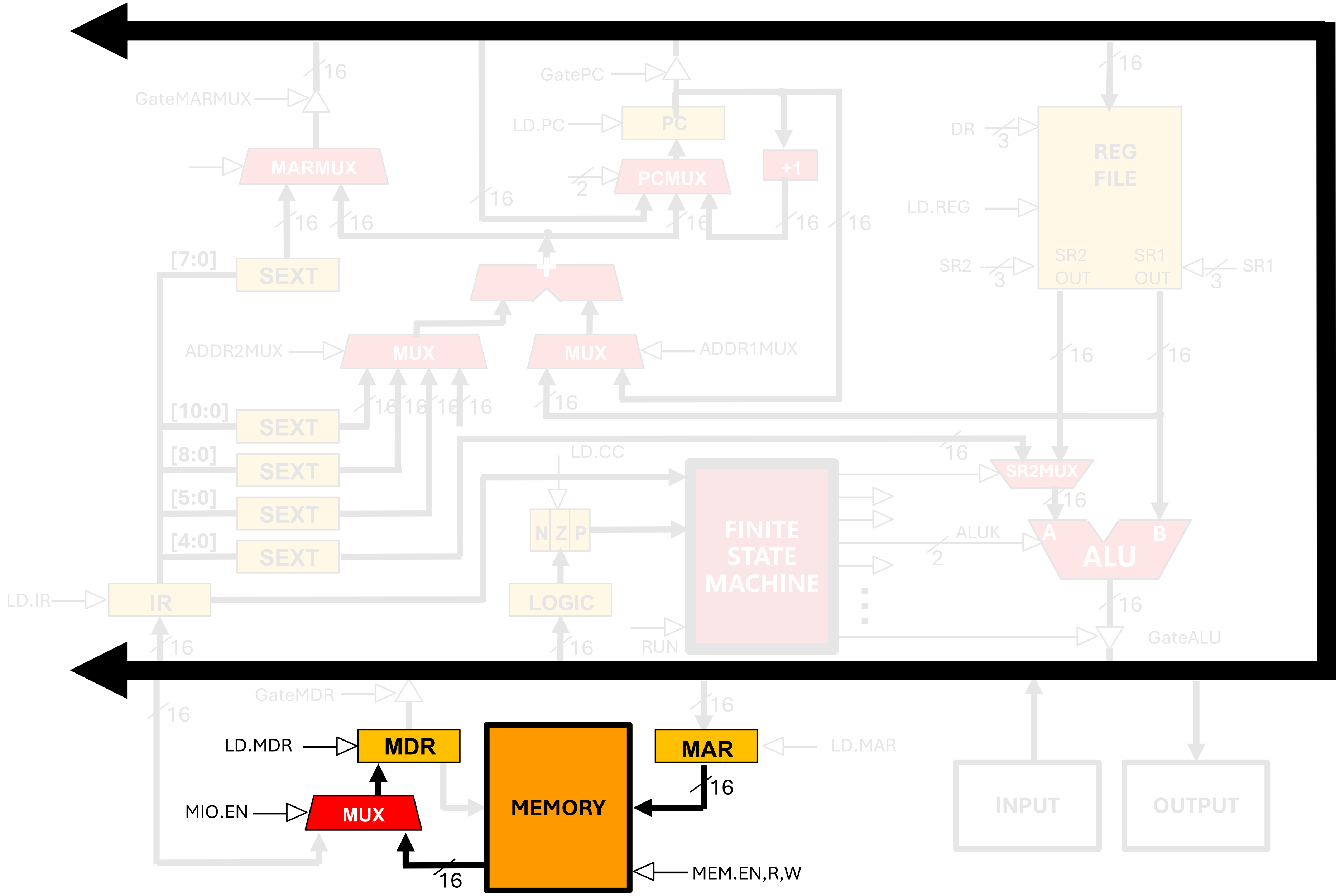
LD (PC-Relative)



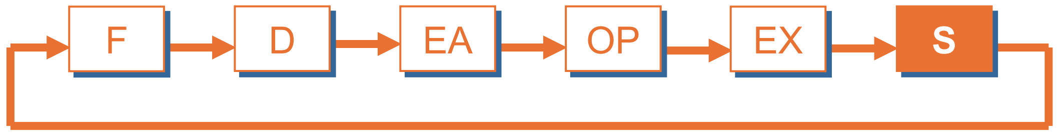


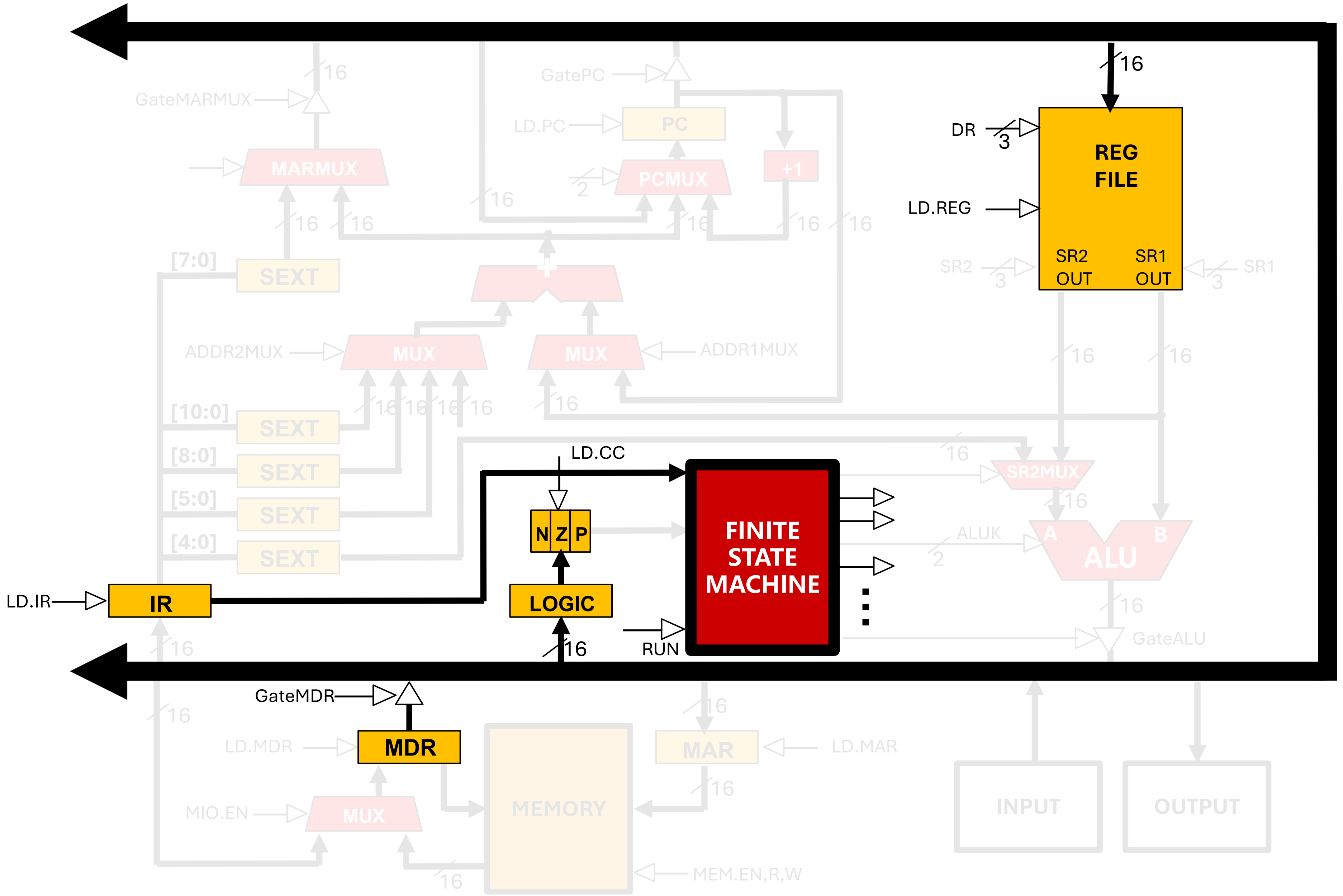
LD (PC-Relative)





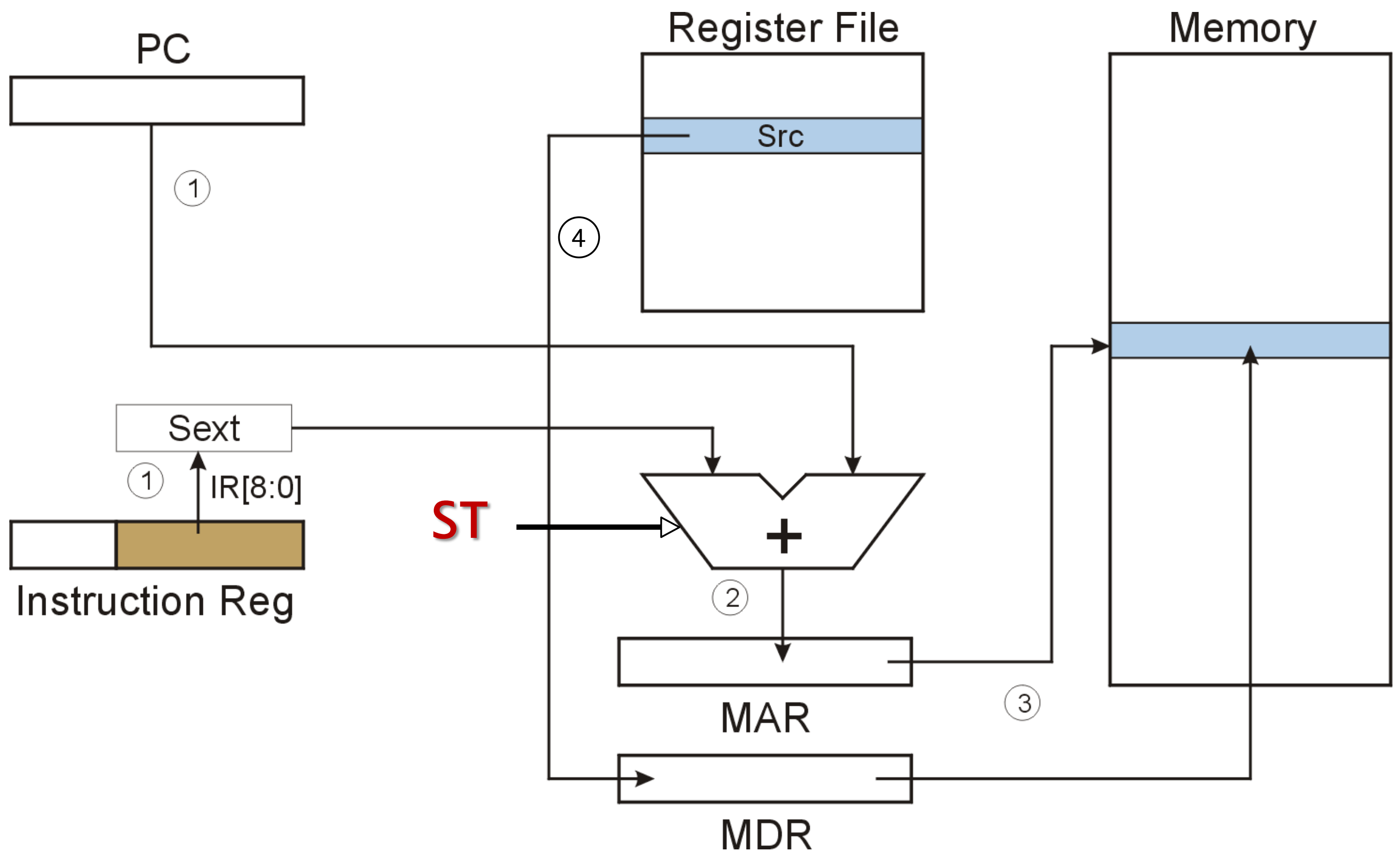
LD (PC-Relative)



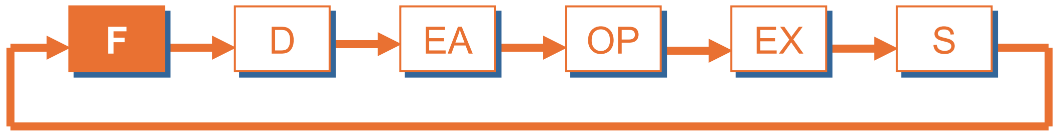


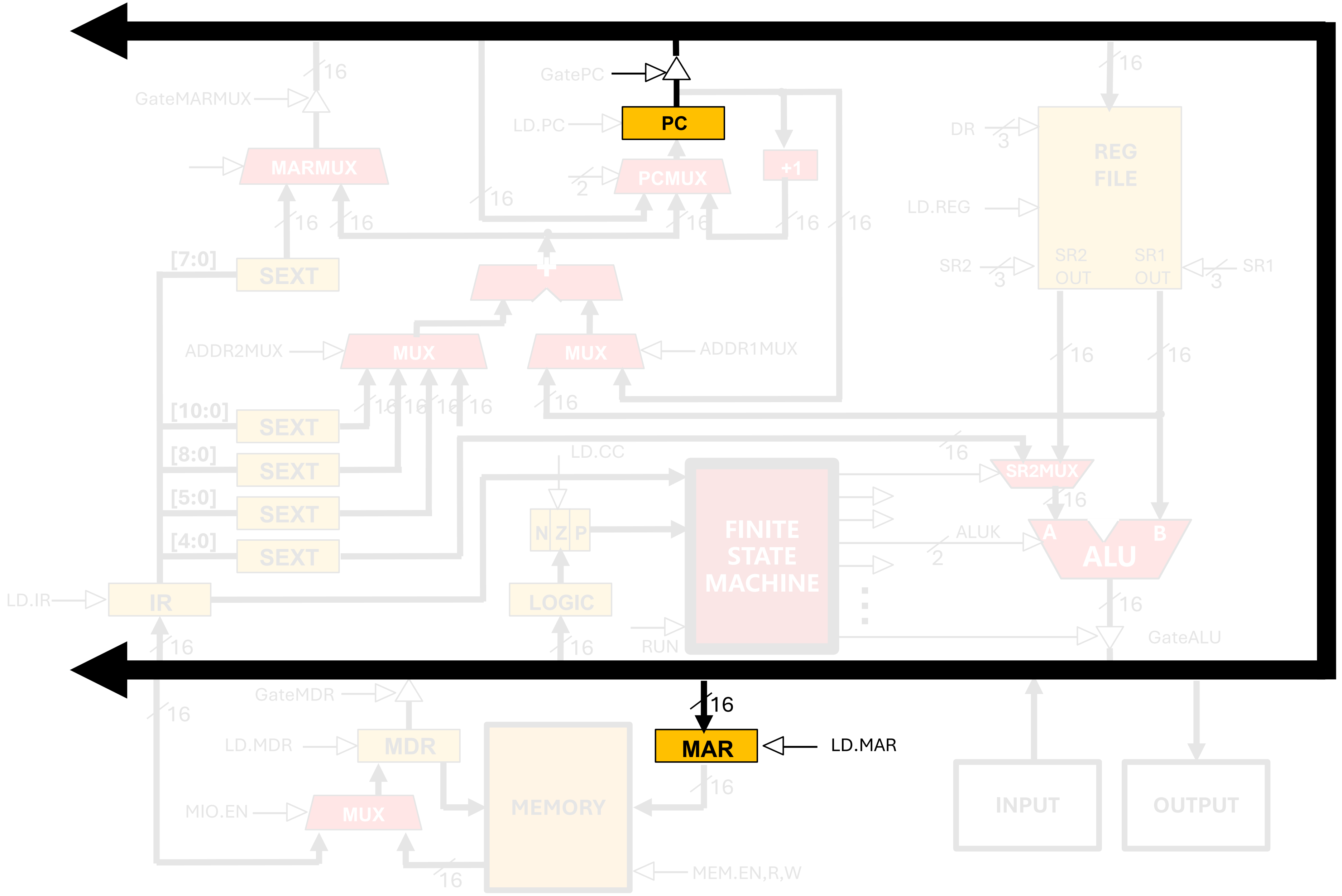
ST (PC-Relative) ST SR, PCoffset9

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

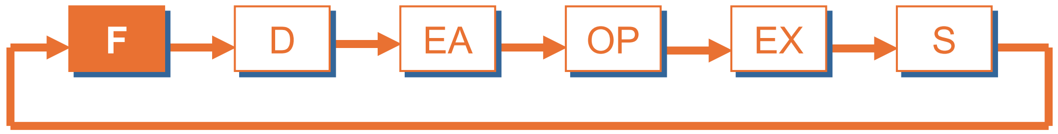


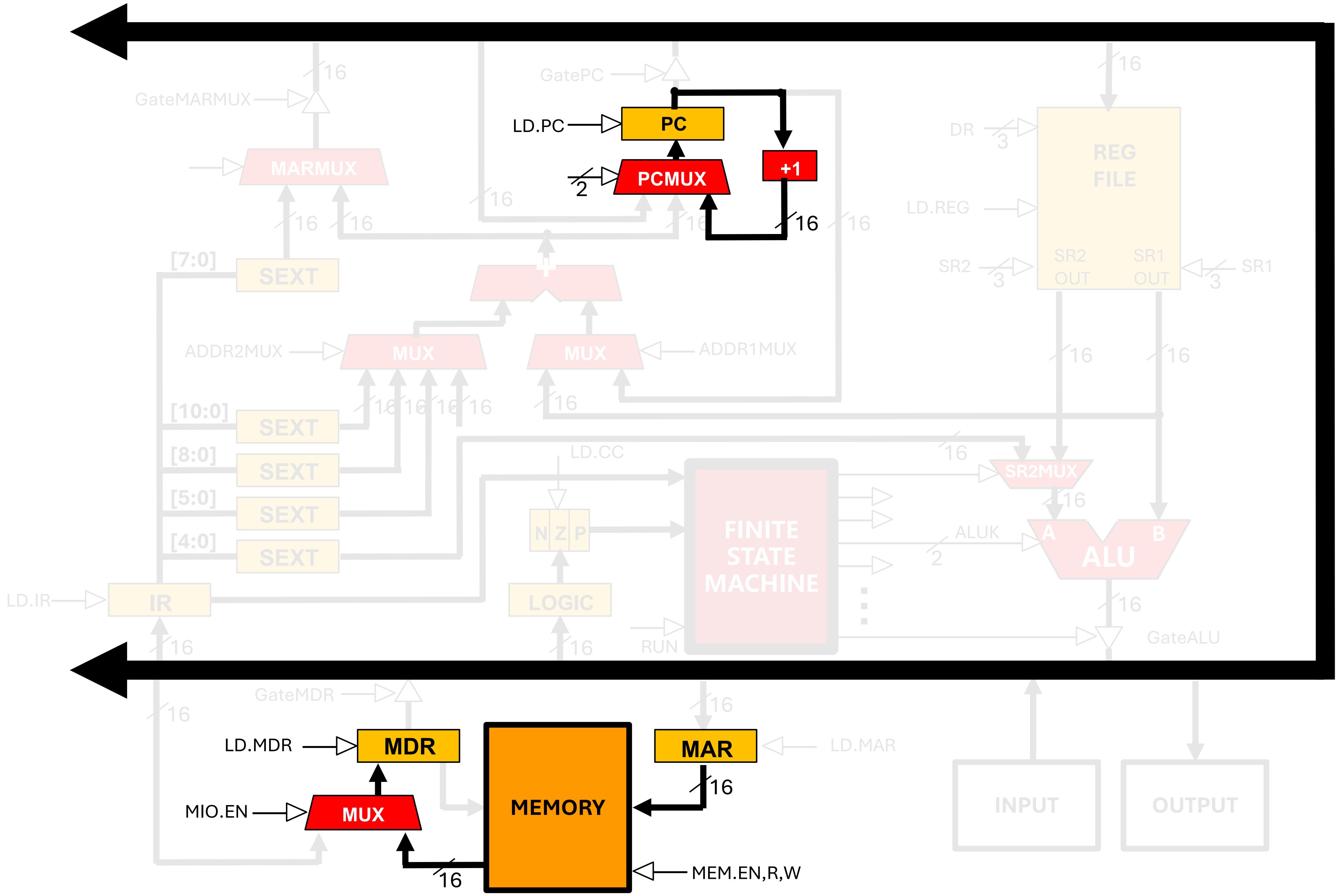
ST (PC-Relative)



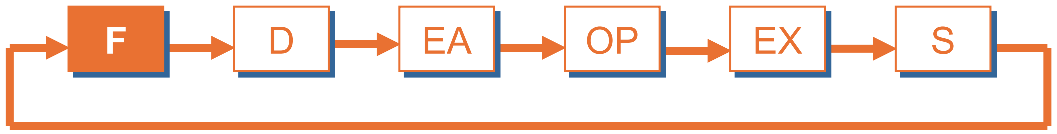


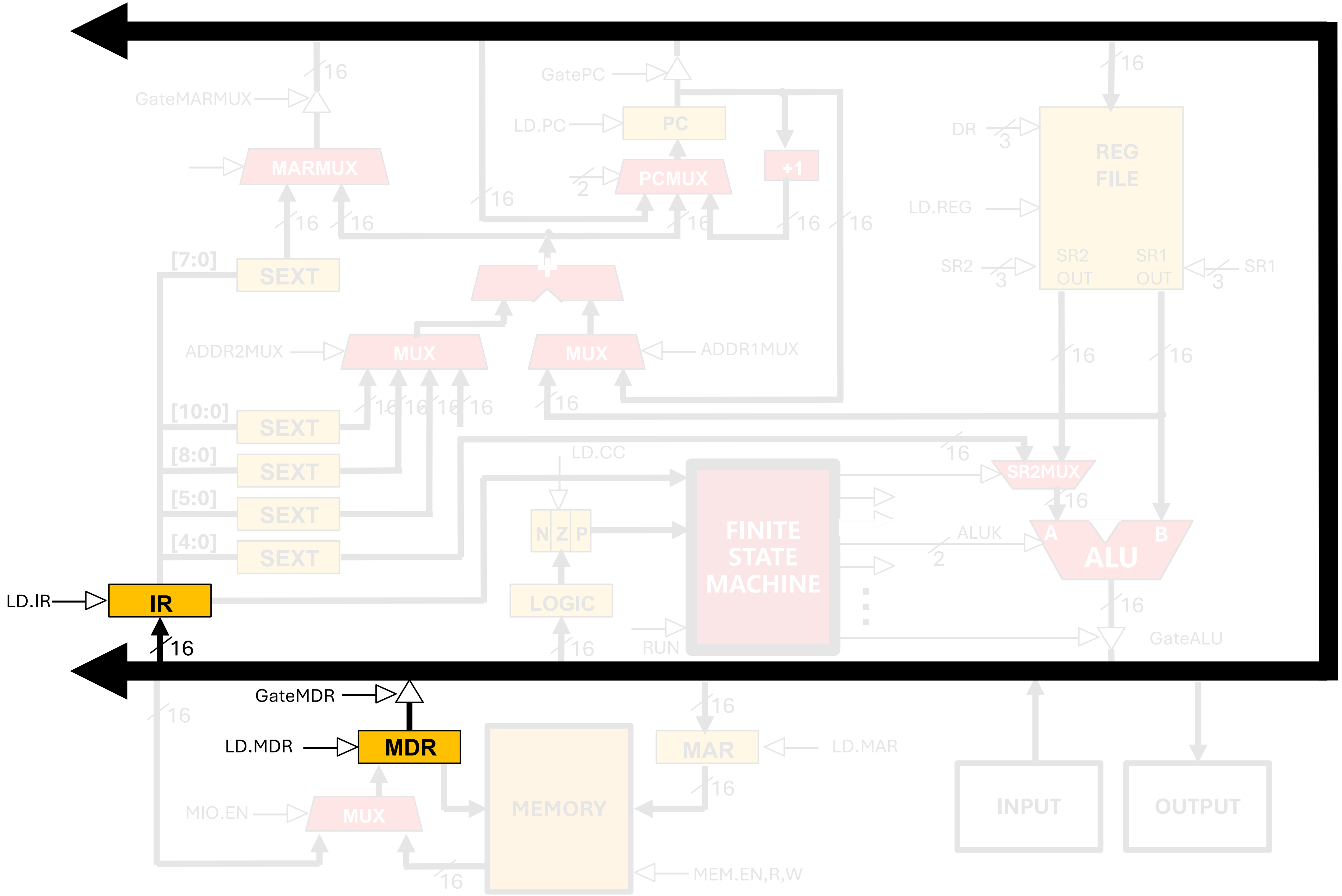
ST (PC-Relative)



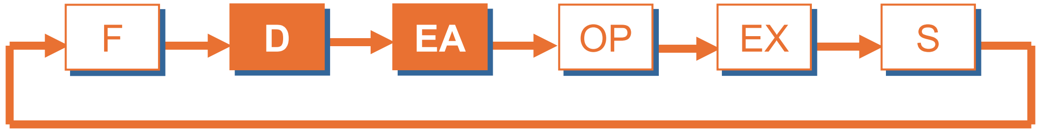


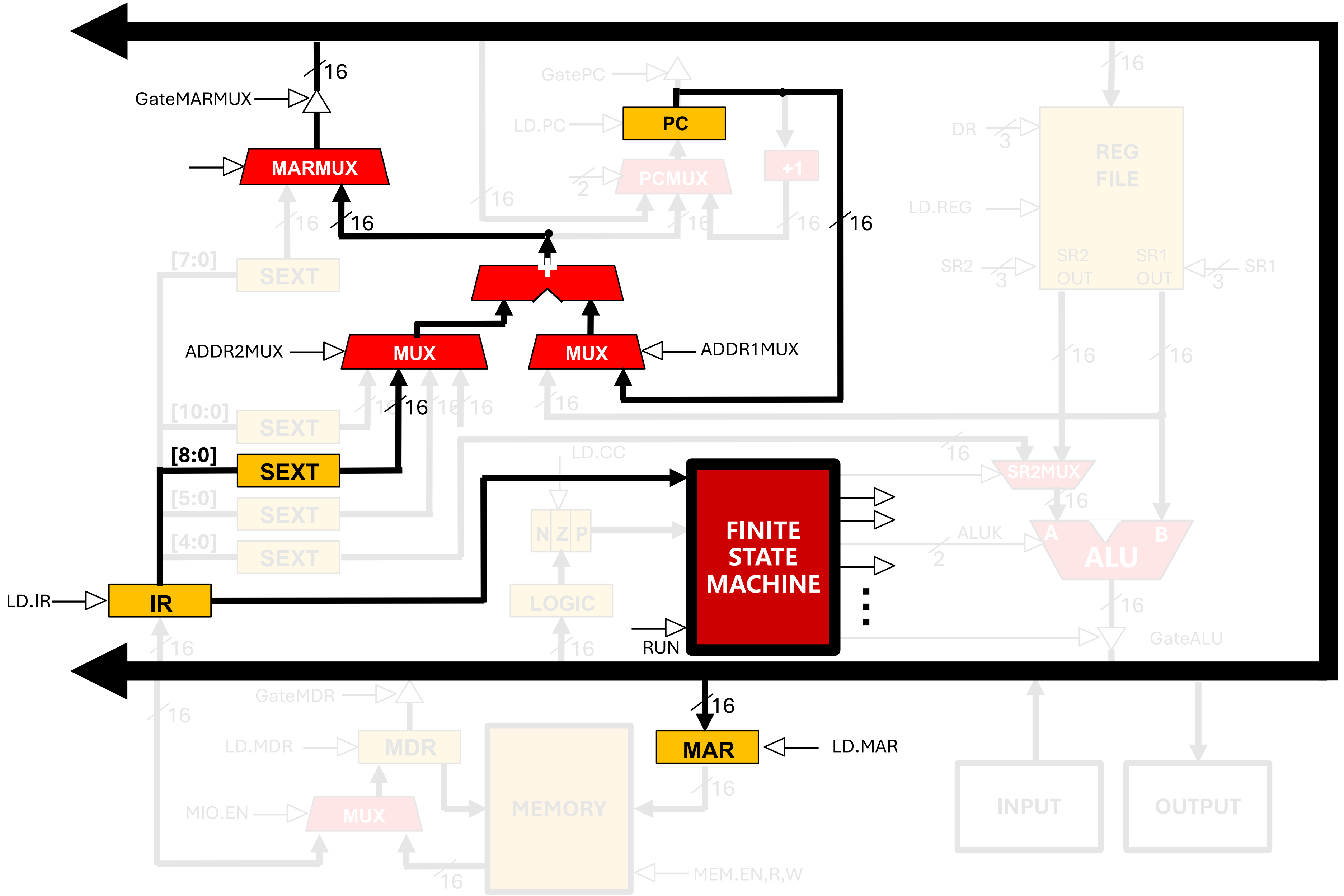
ST (PC-Relative)



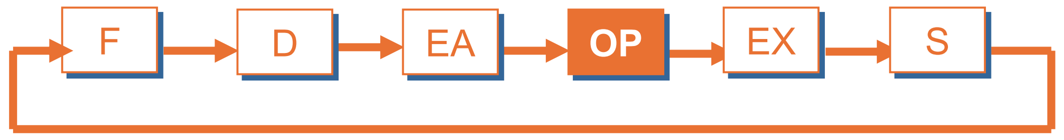


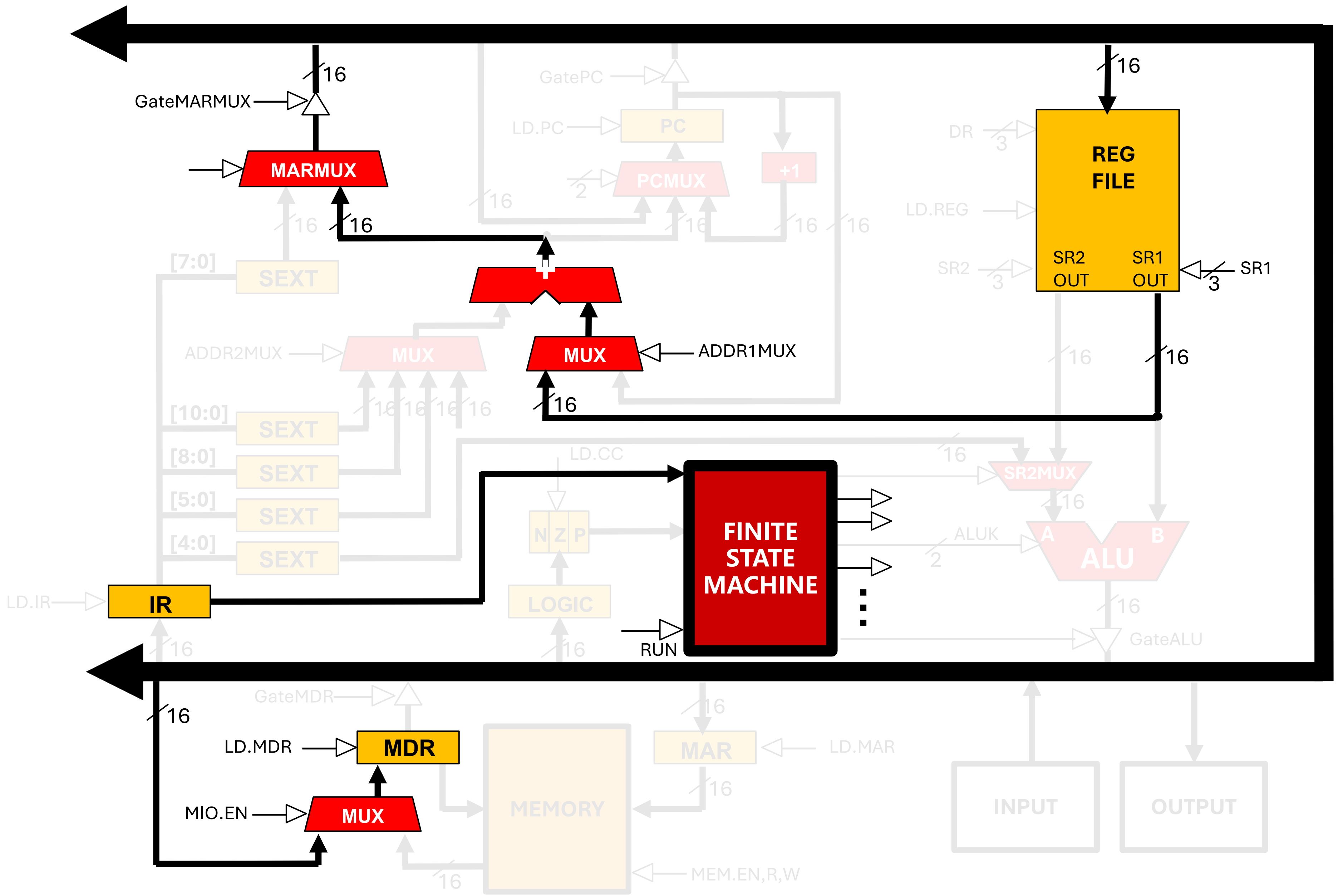
ST (PC-Relative)



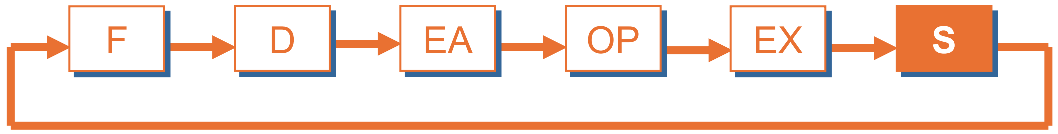


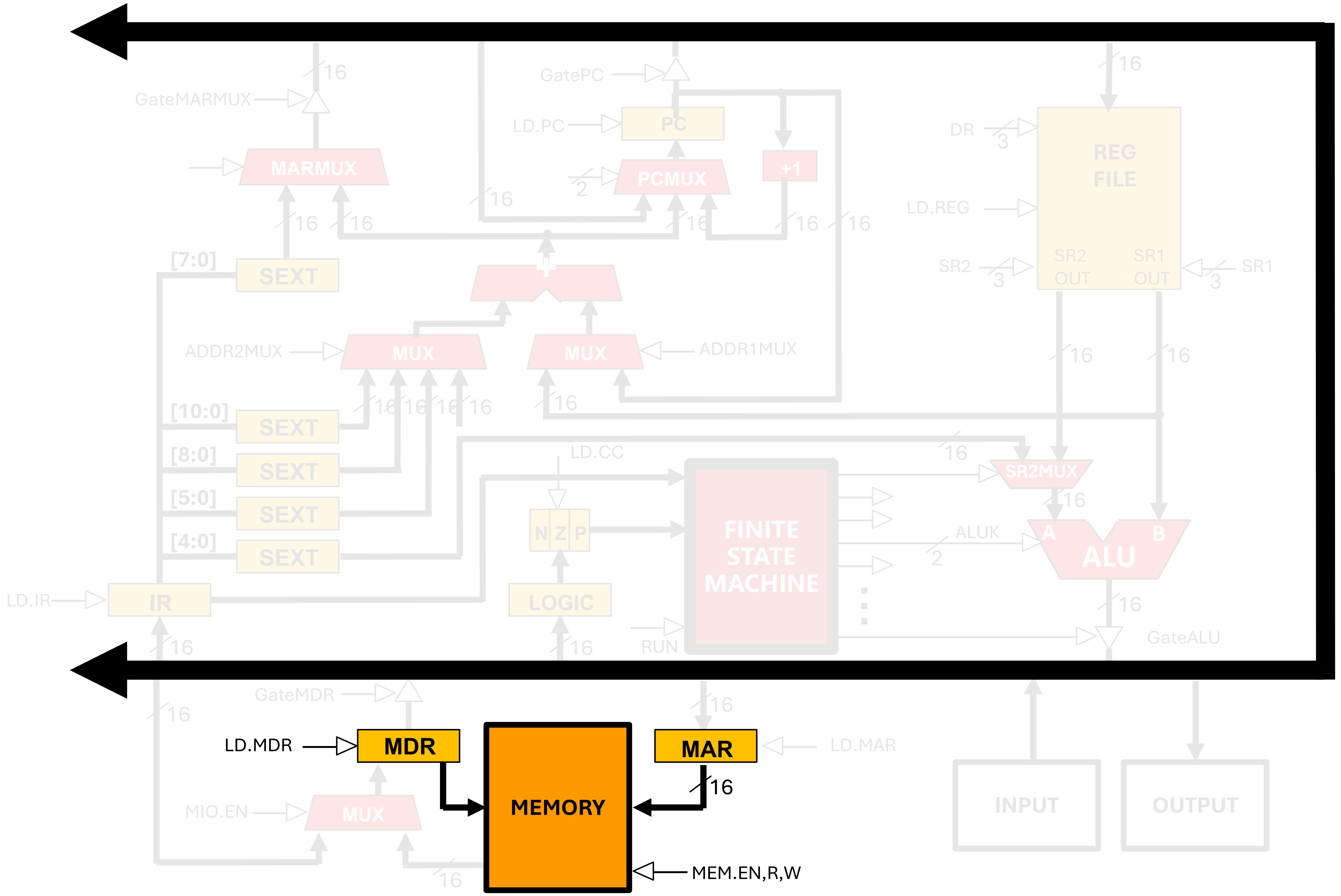
ST (PC-Relative)





ST (PC-Relative)





LC-3 Data Movement Instructions

LC-3 PC-Relative Load/Store

LC-3 Indirect, Base+offset Load/Store

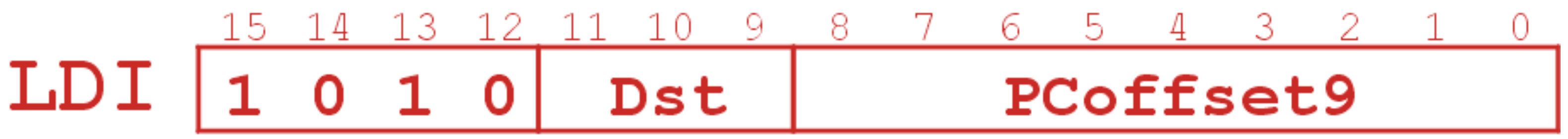
Indirect Addressing Mode

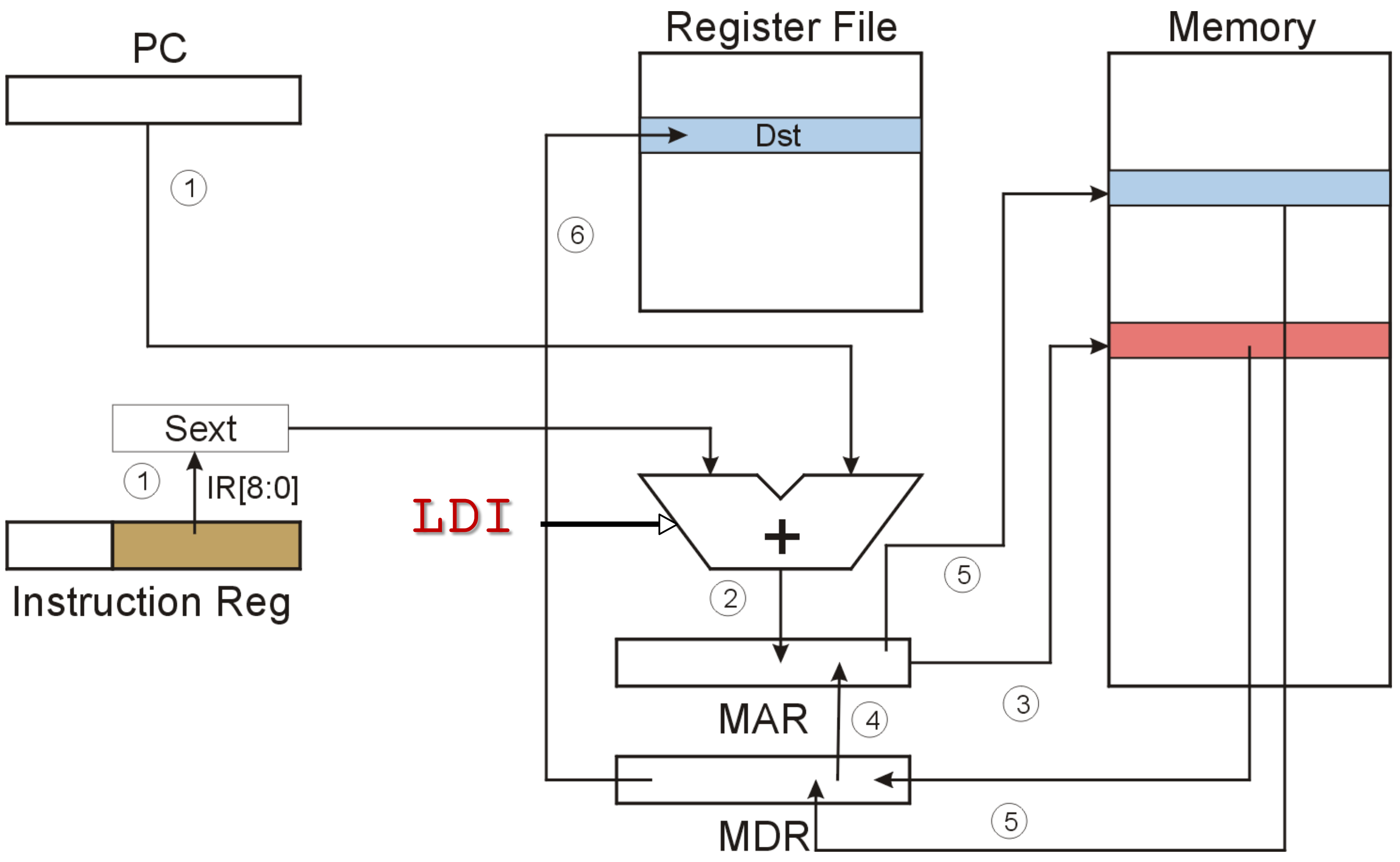
- PC-relative mode, can only address data within 256 words of the instruction
  - What about the rest of memory?

- Solution1 → **Read address from memory location, then load/store to that address**

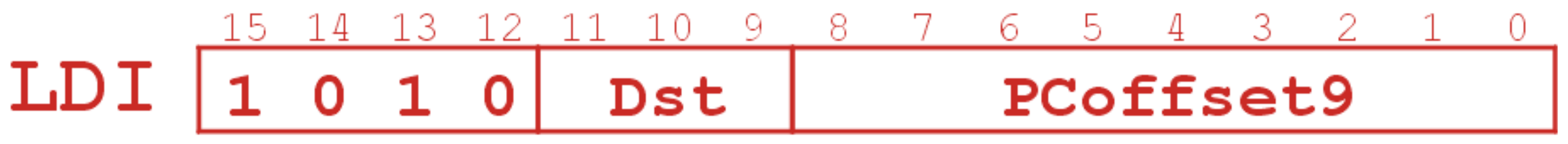
- First address is generated from PC and IR

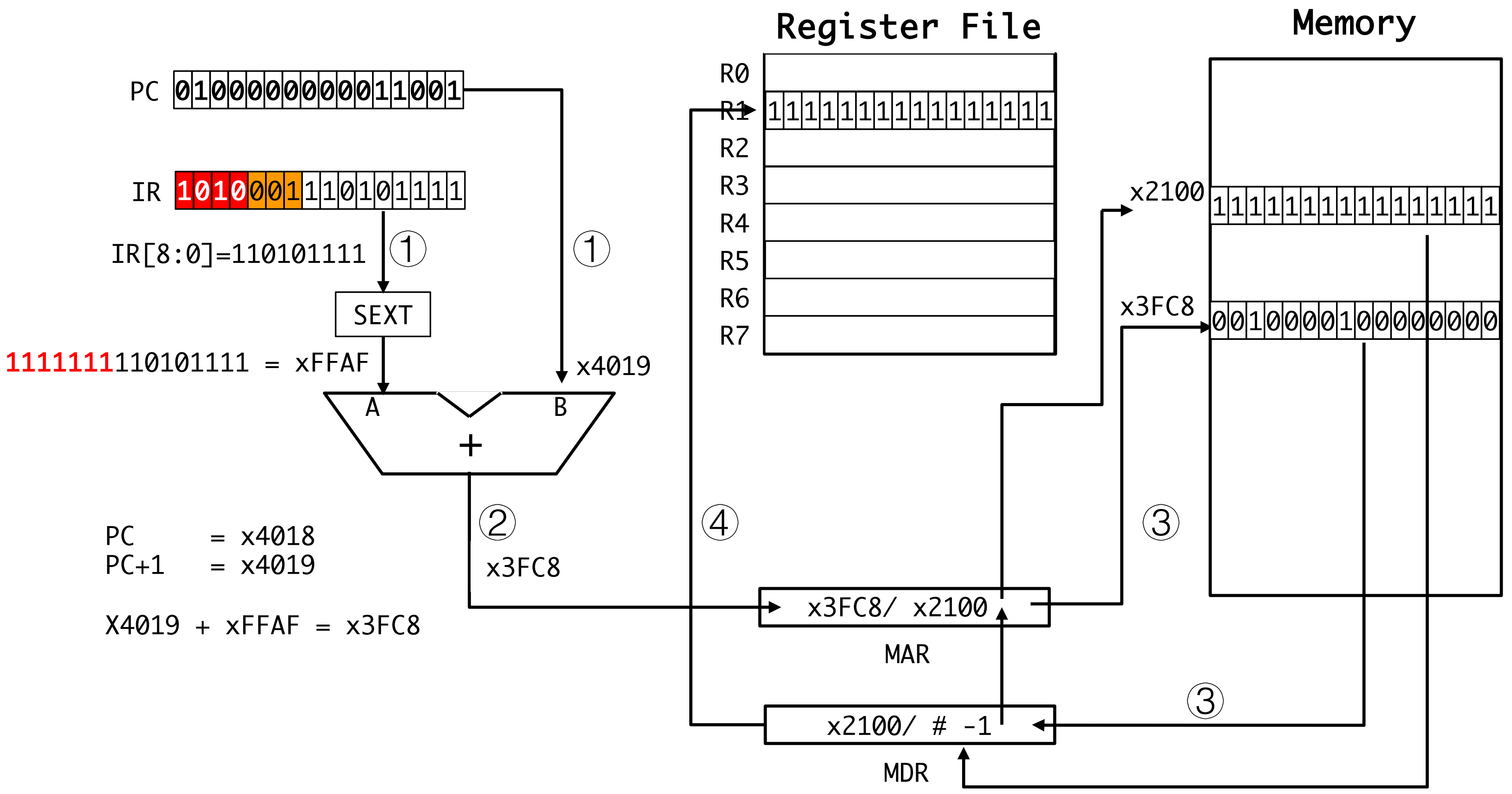
LDI (Indirect) LDI DR, PCoffset9





LDI (Indirect) : LDI R1, x1AF

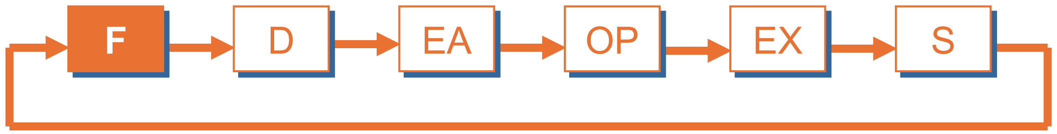


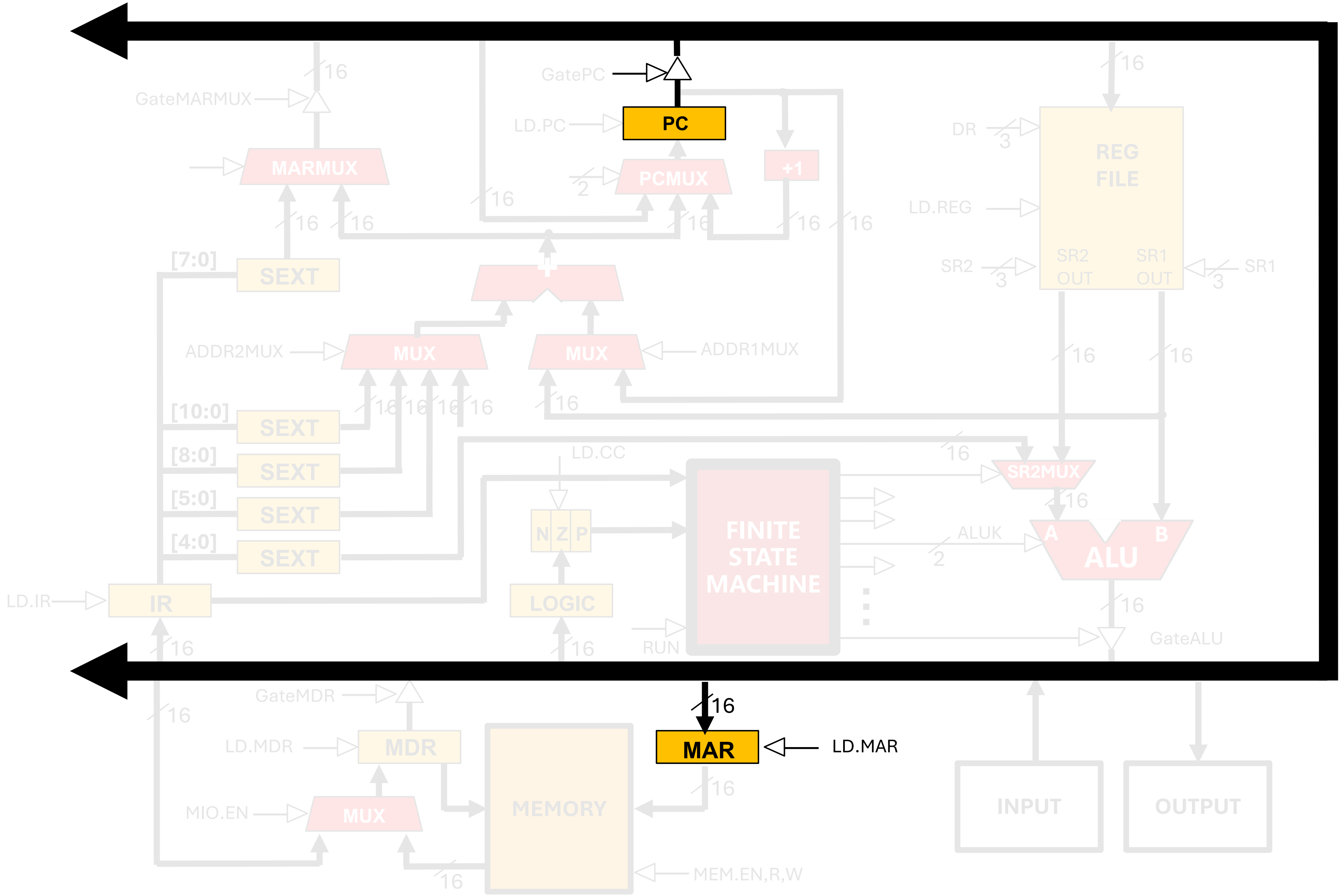


PC = x4018
PC+1 = x4019

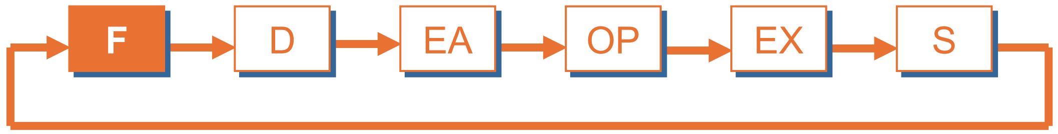
X4019 + xFFAF = x3FC8

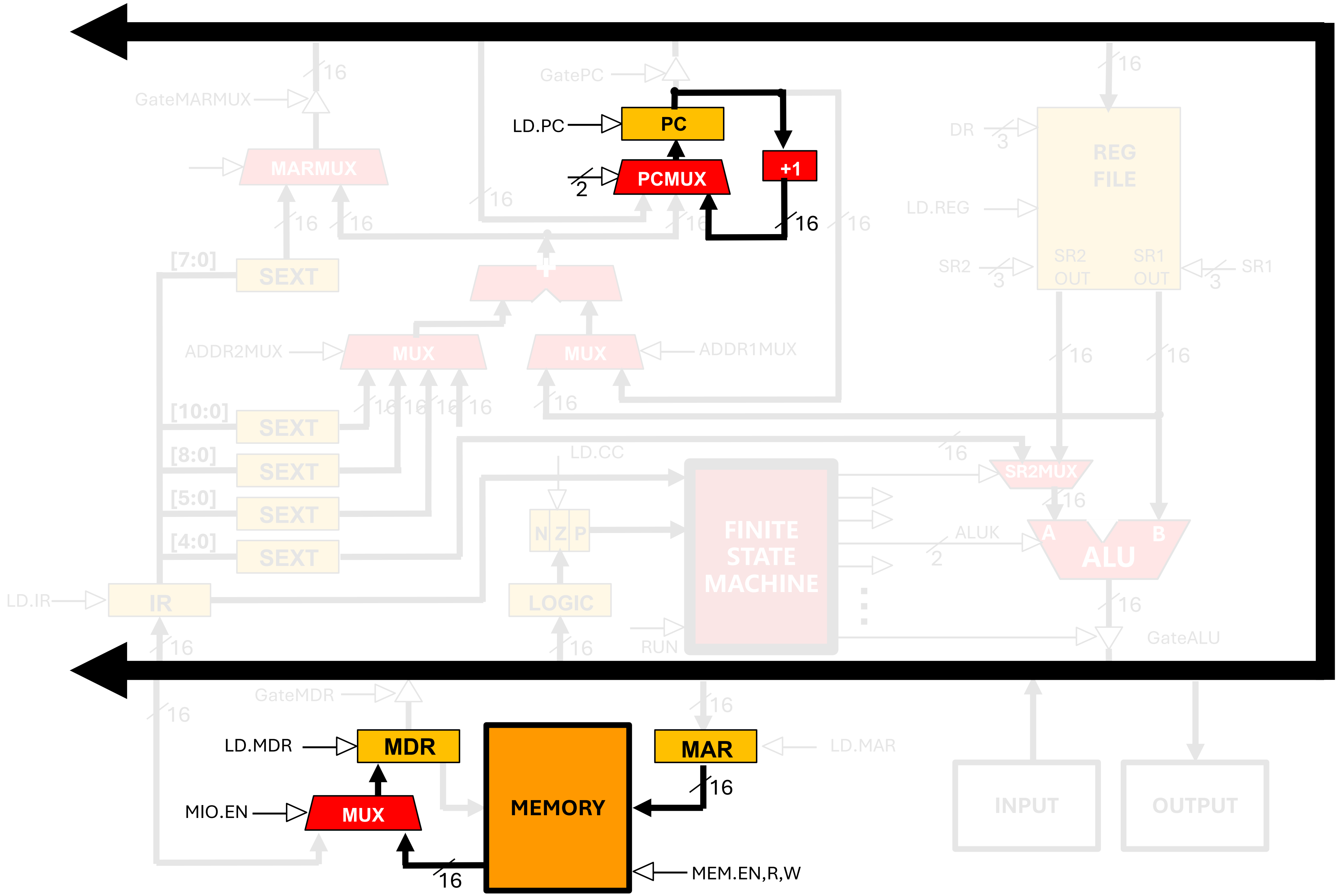
LDI (Indirect)



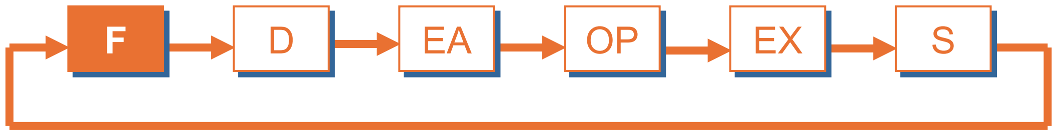


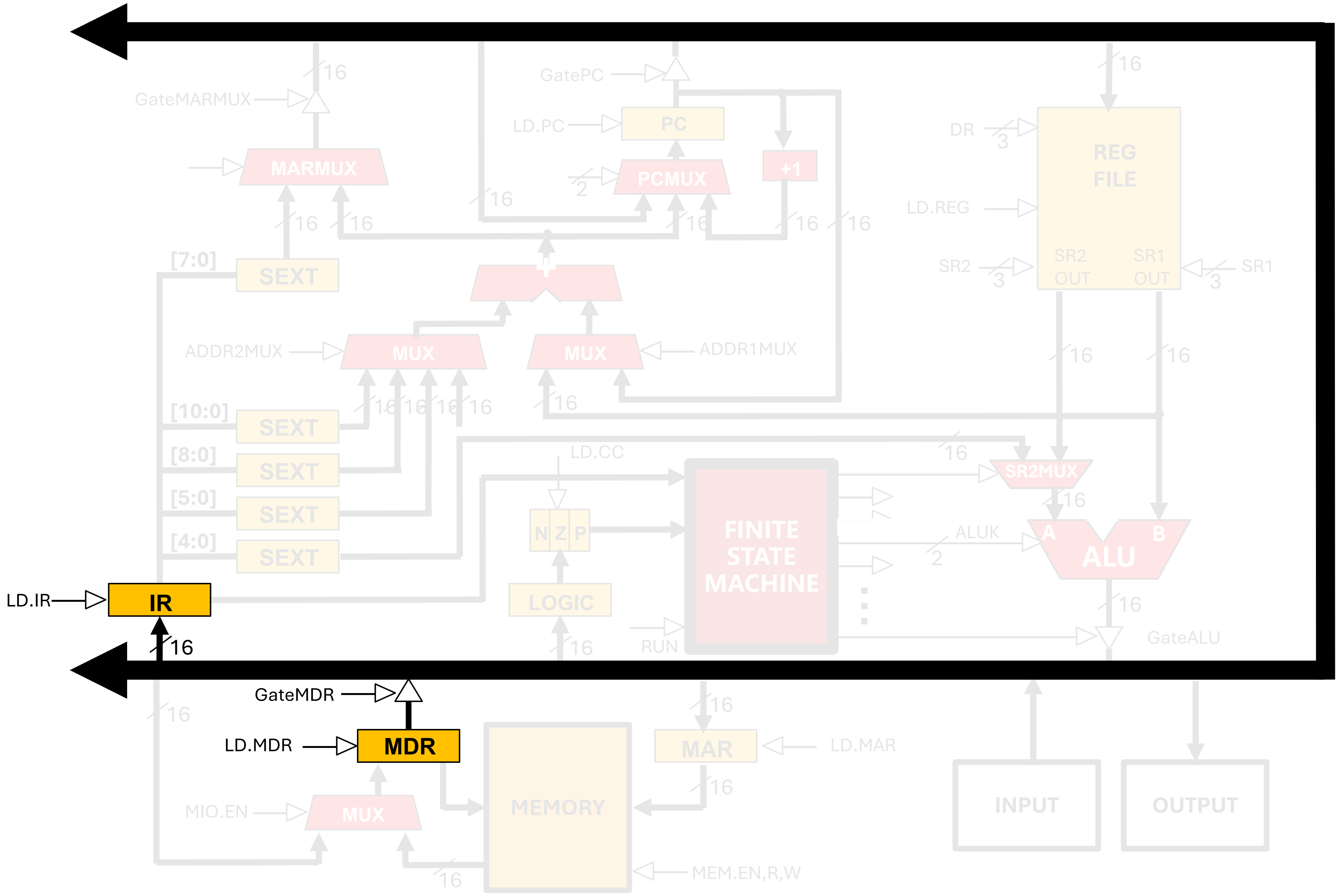
LDI (Indirect)



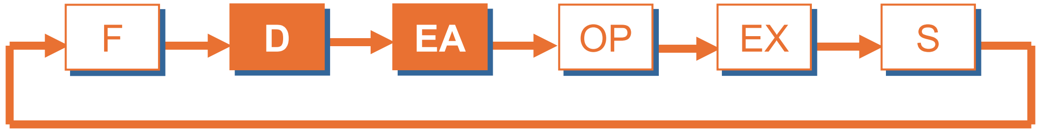


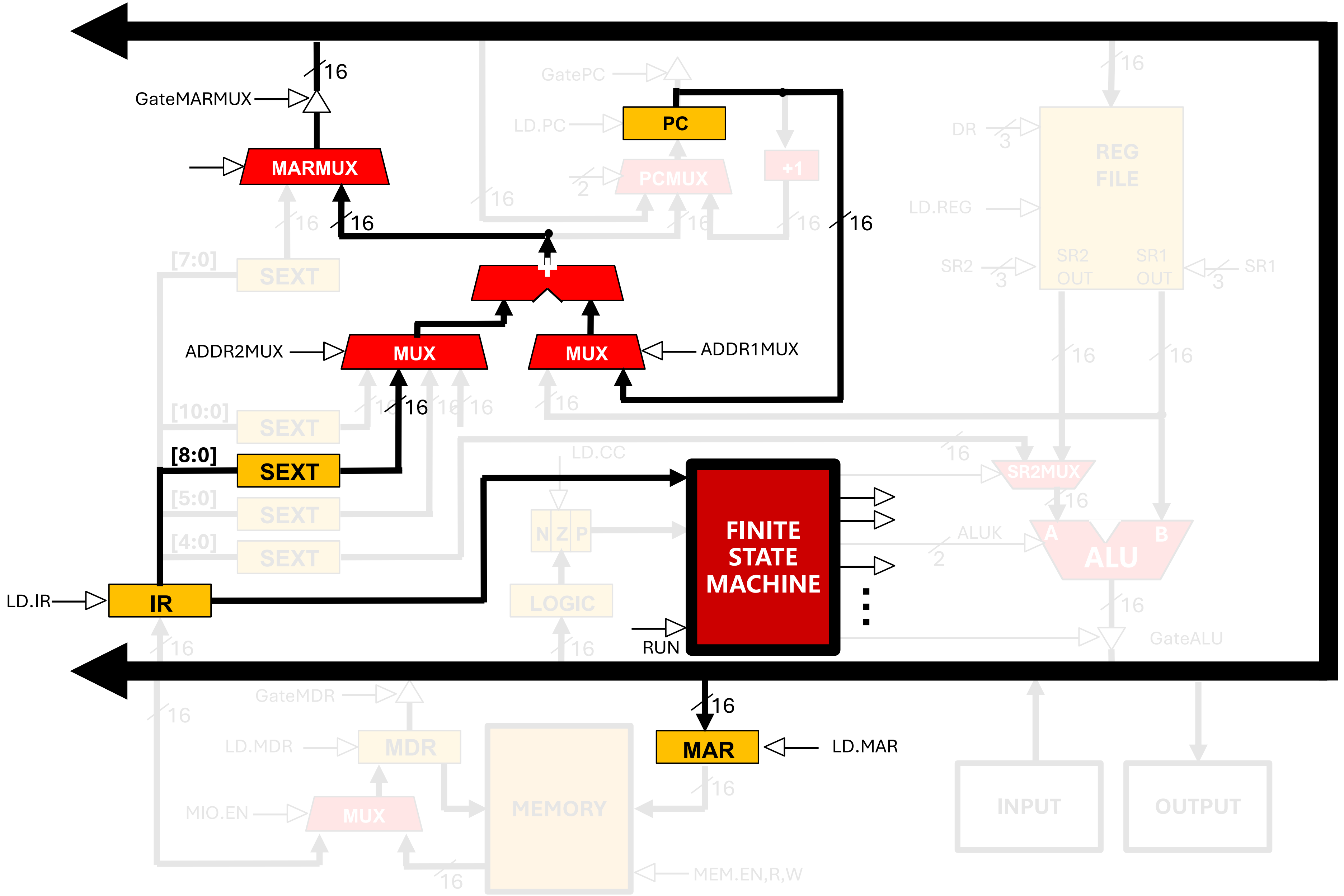
LDI (Indirect)



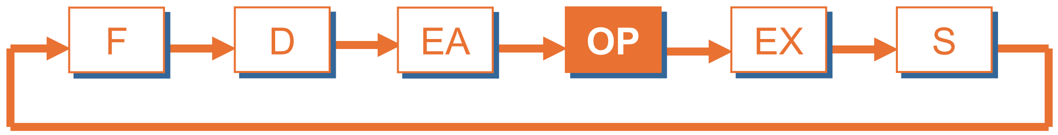


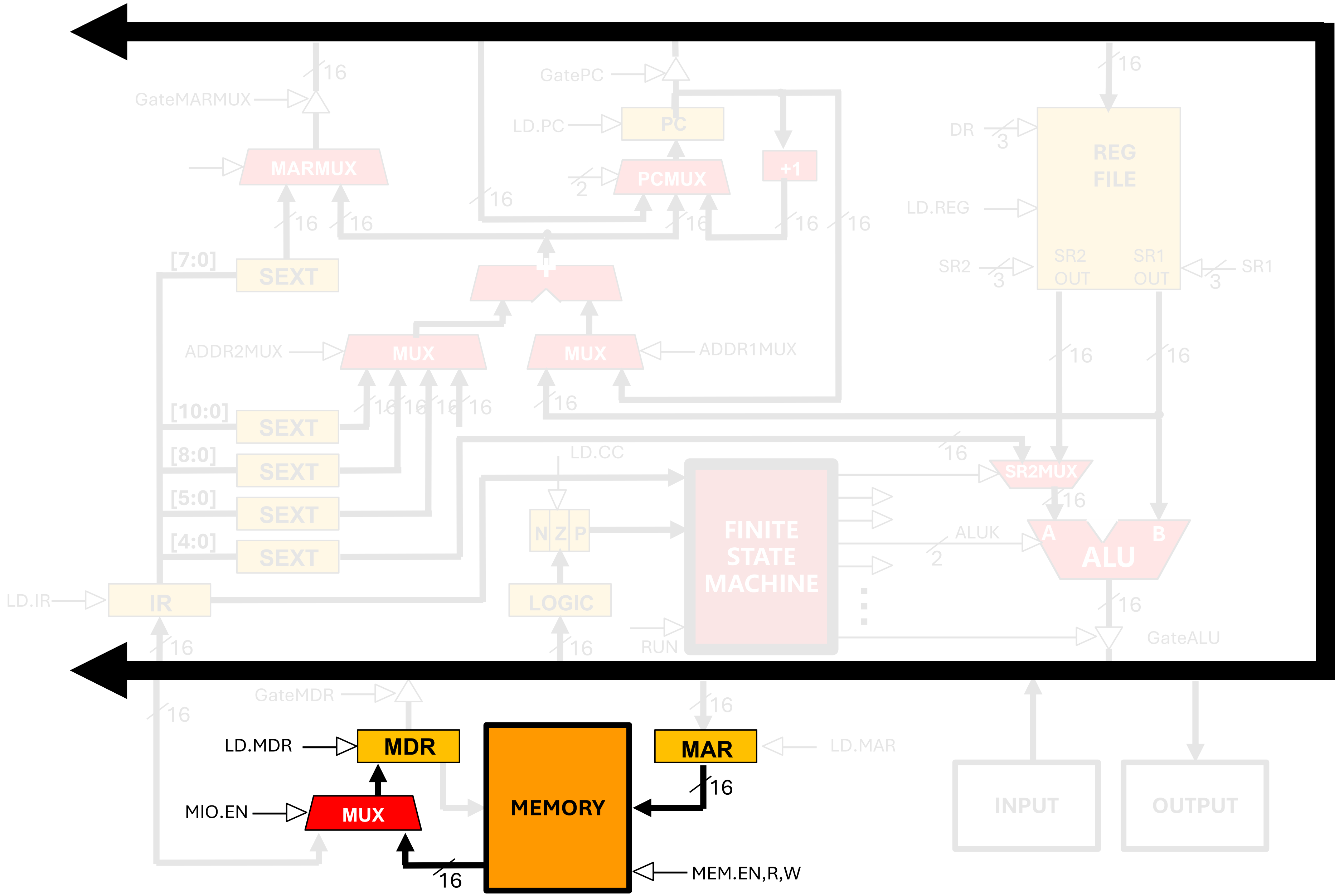
LDI (Indirect)



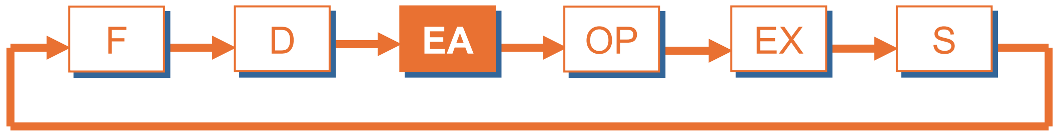


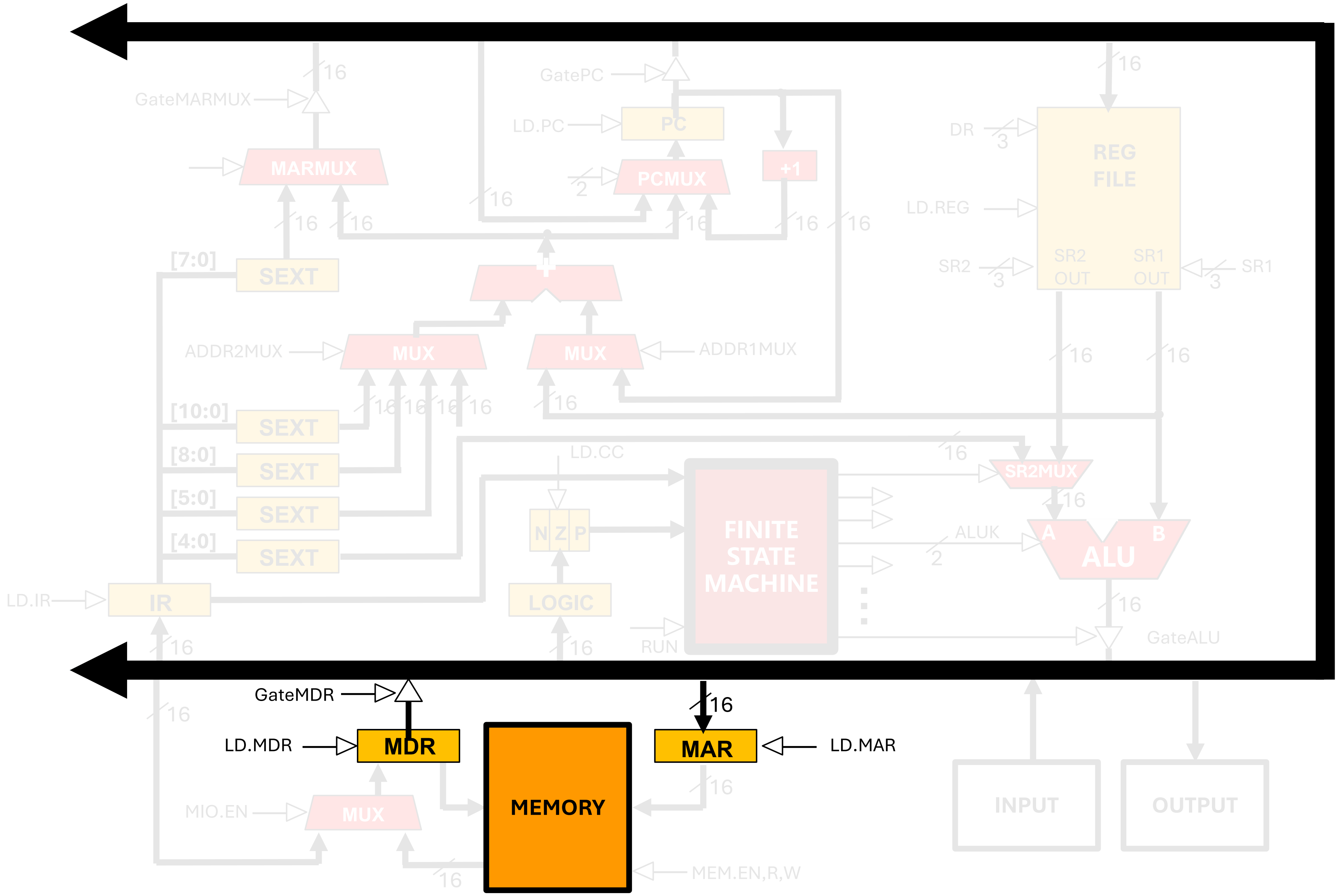
LDI (Indirect)



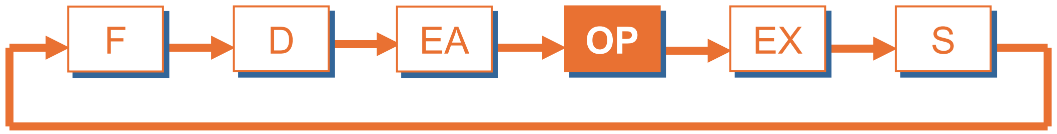


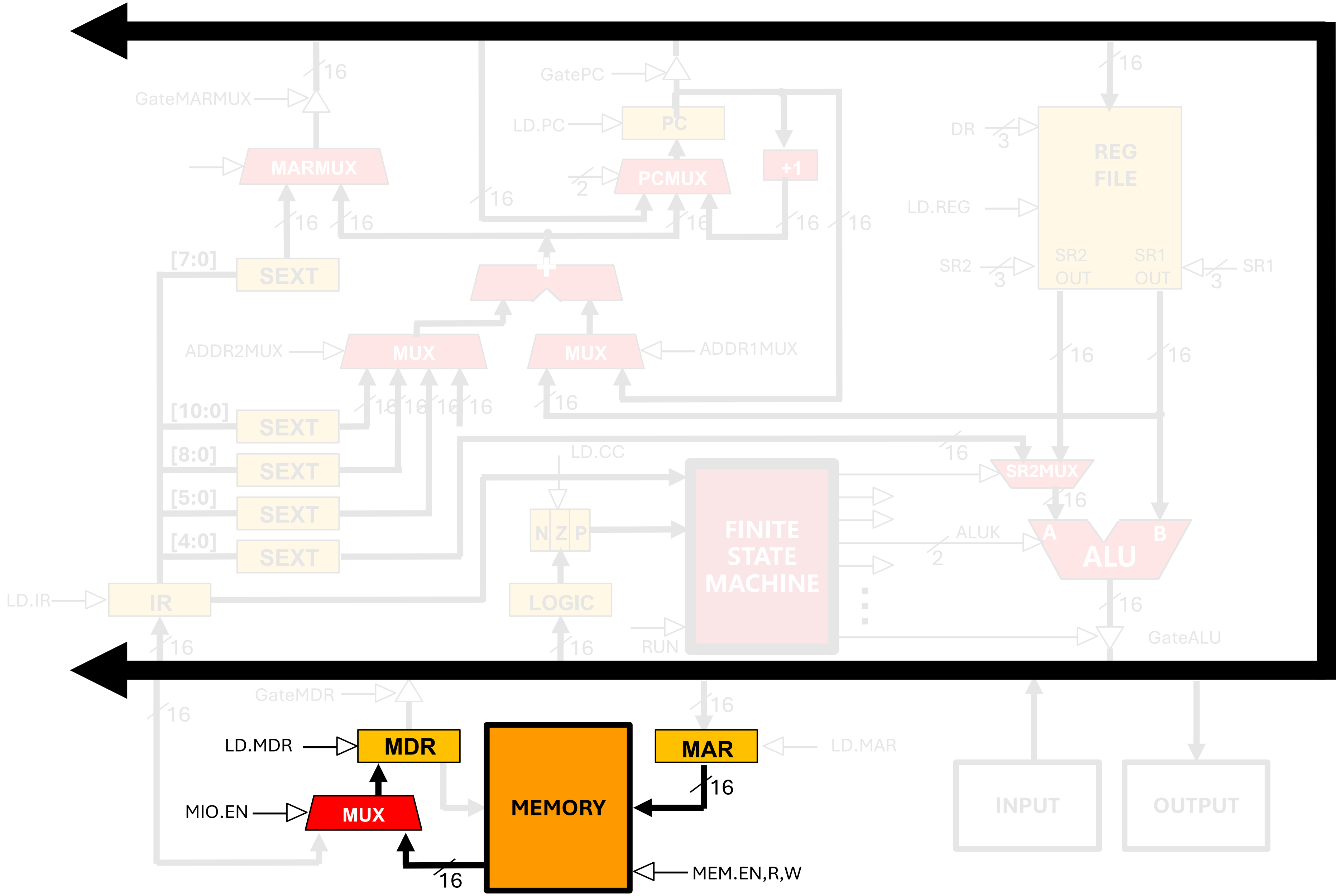
LDI (Indirect)



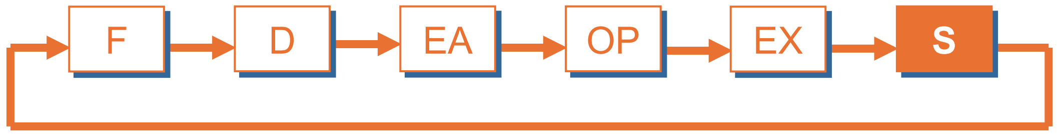


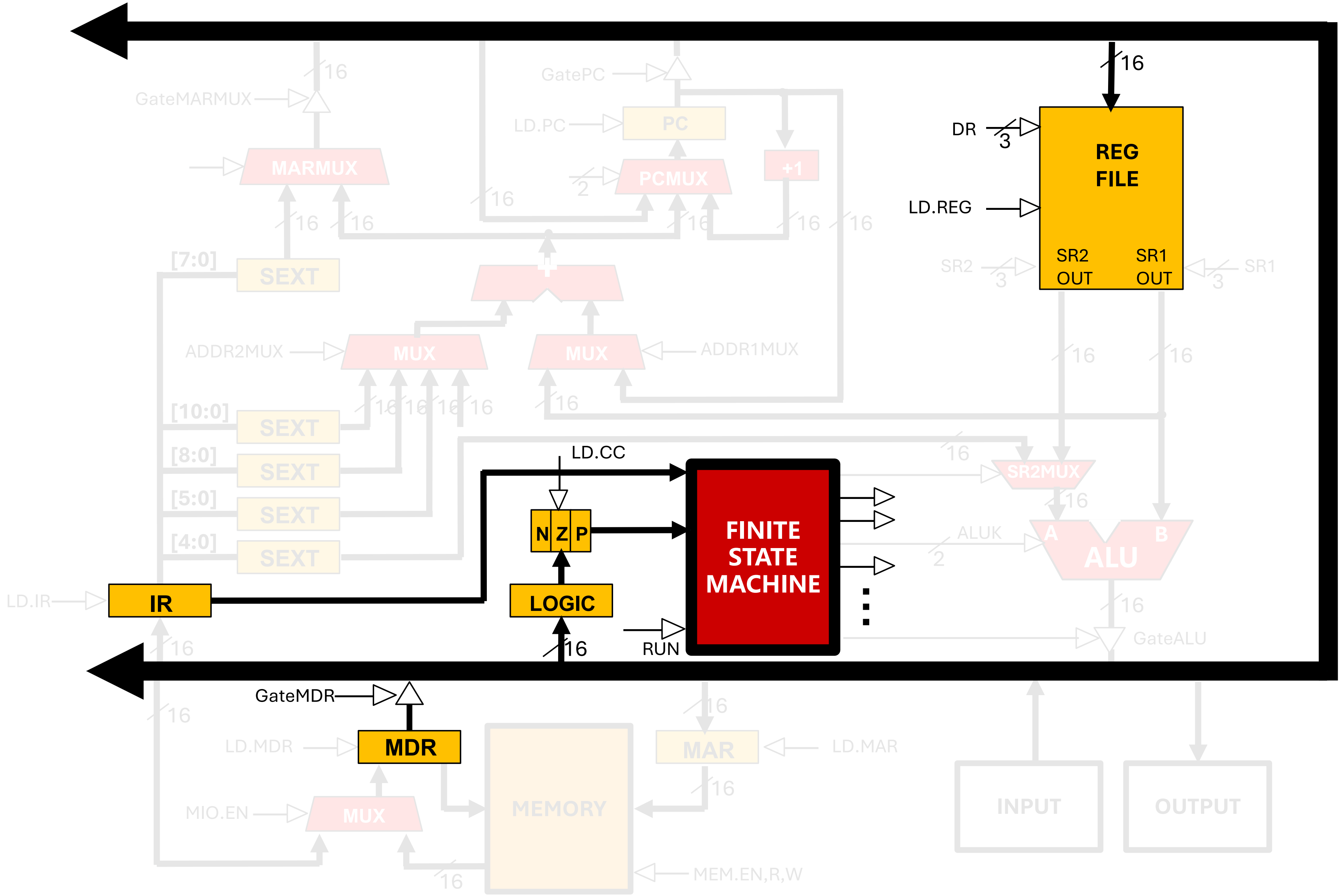
LDI (Indirect)





LDI (Indirect)





Base + Offset Addressing Mode

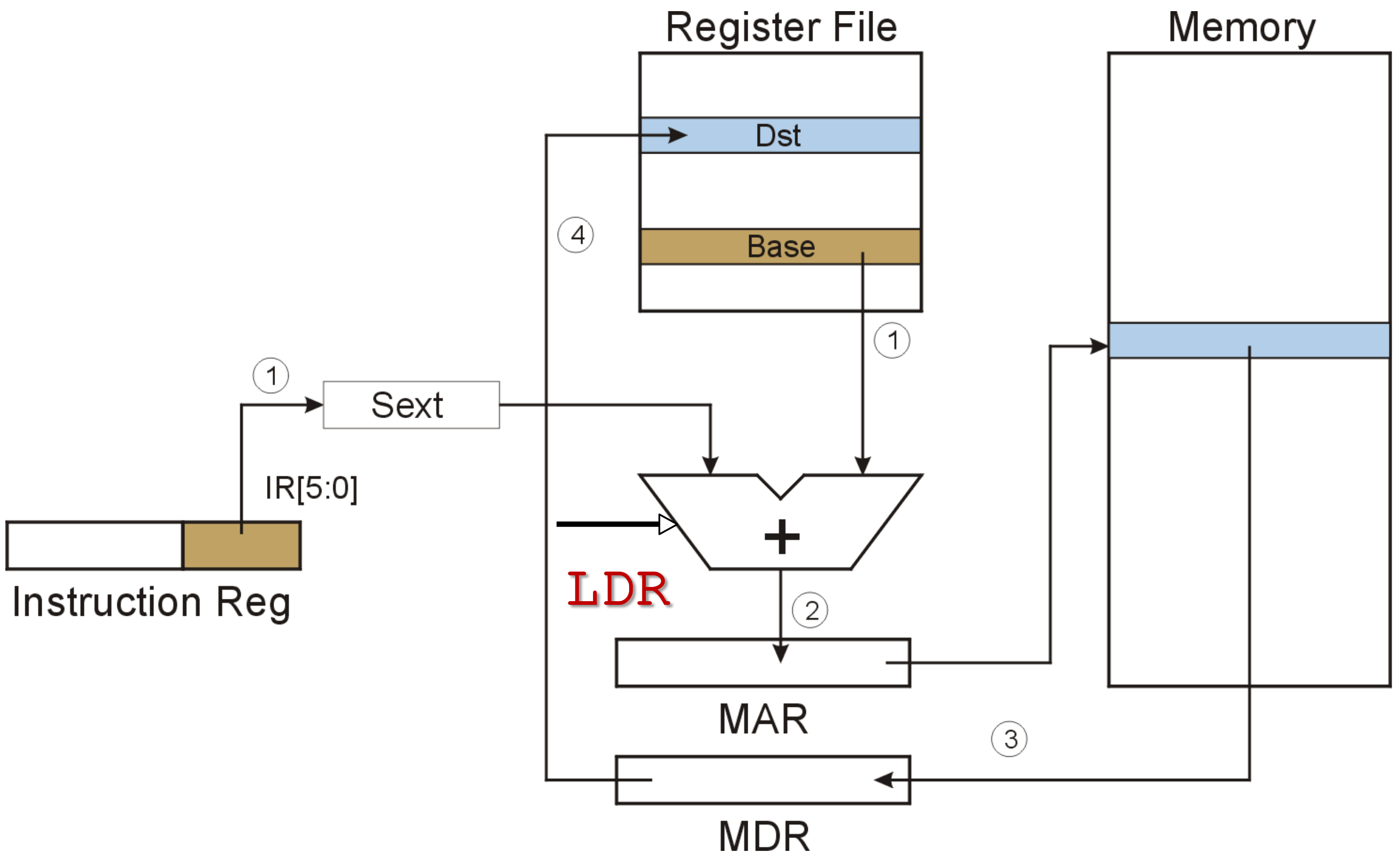
- PC-relative mode, can only address data within 256 words of the instruction
  - What about the rest of memory?

- Solution2 → Use a register to generate a full 16-bit address

- 4 bits for opcode, 3 for src/dest register,
- 3 bits for *base* register -- remaining 6 bits are used as a *signed offset*
  - Offset is *sign-extended* before adding to base register

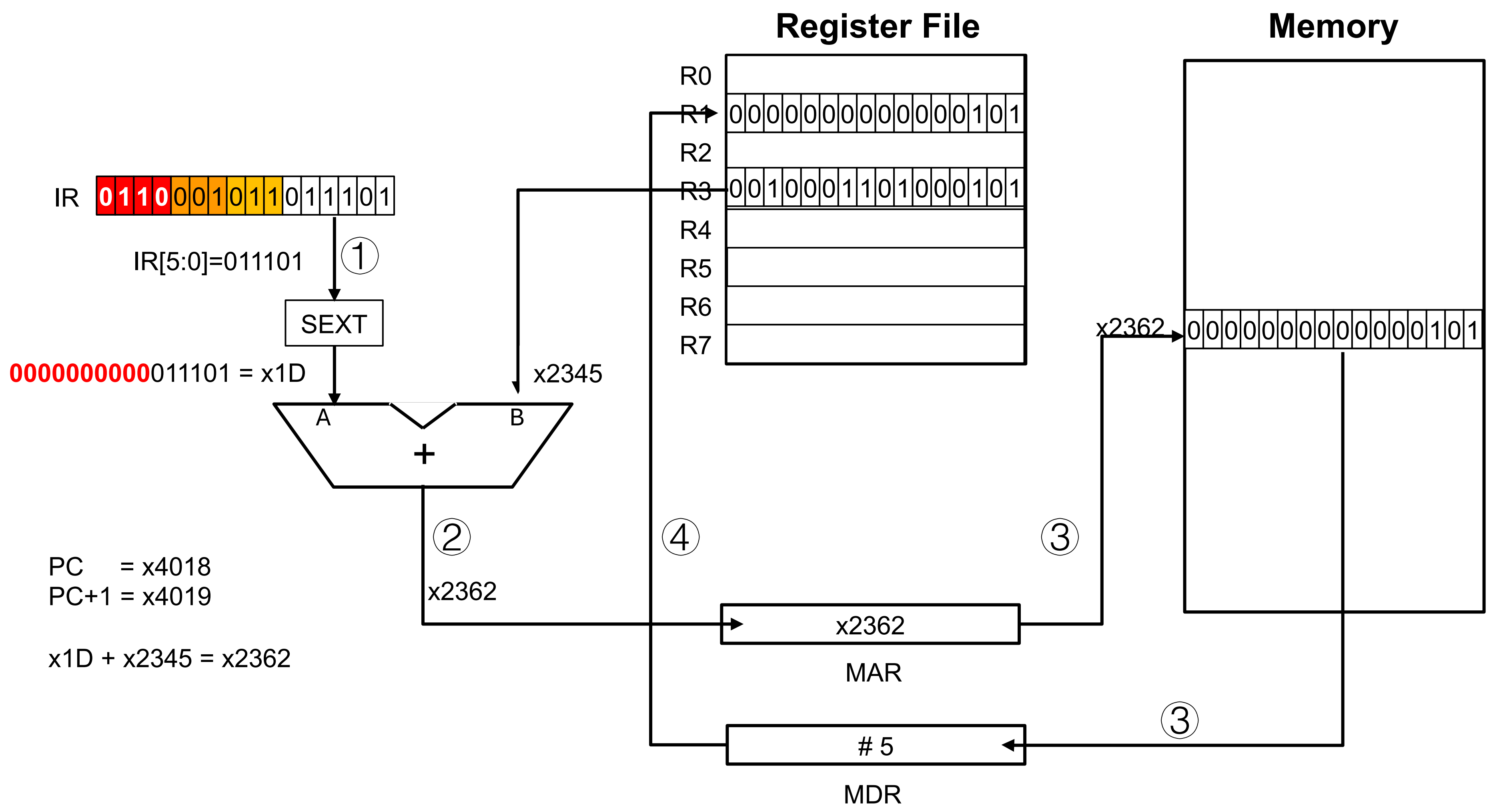
LDR (Base+Offset) LDR DR, BaseR, offset6

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



LDR (Base+Offset) : LD R1, R3, x1D

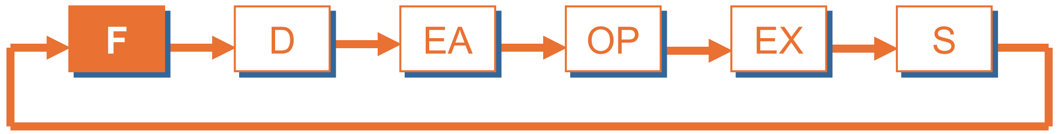
	15	14	13	12	11 10 9	8 7 6	5 4 3 2 1 0
LDR	0	1	1	0	Dst	Base	offset6

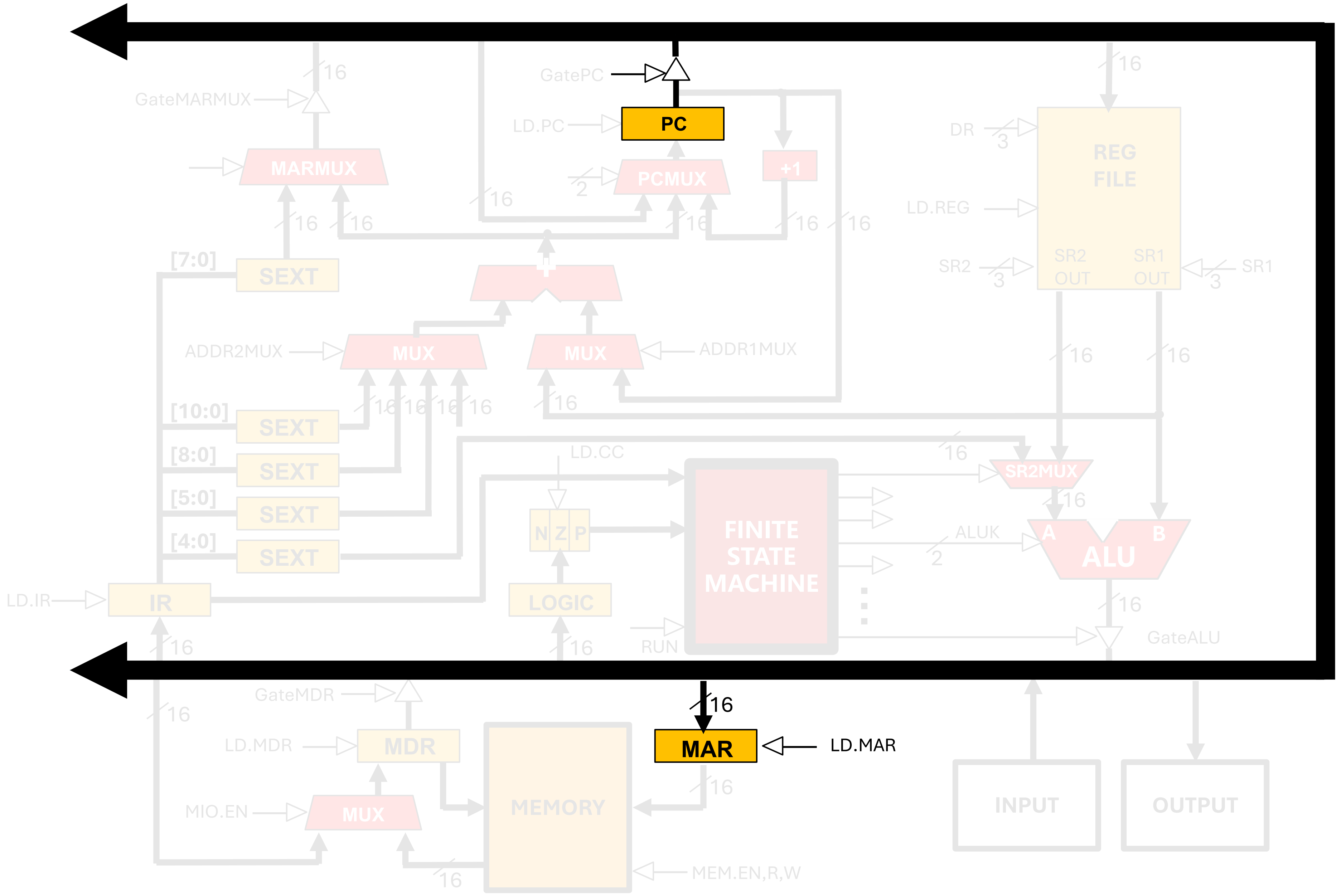


PC   = x4018
PC+1 = x4019

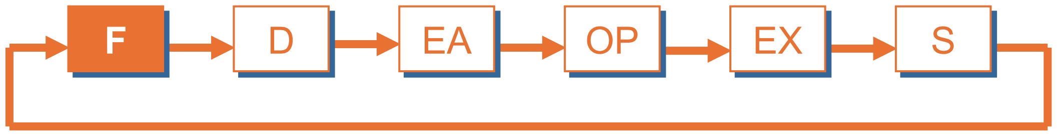
x1D + x2345 = x2362

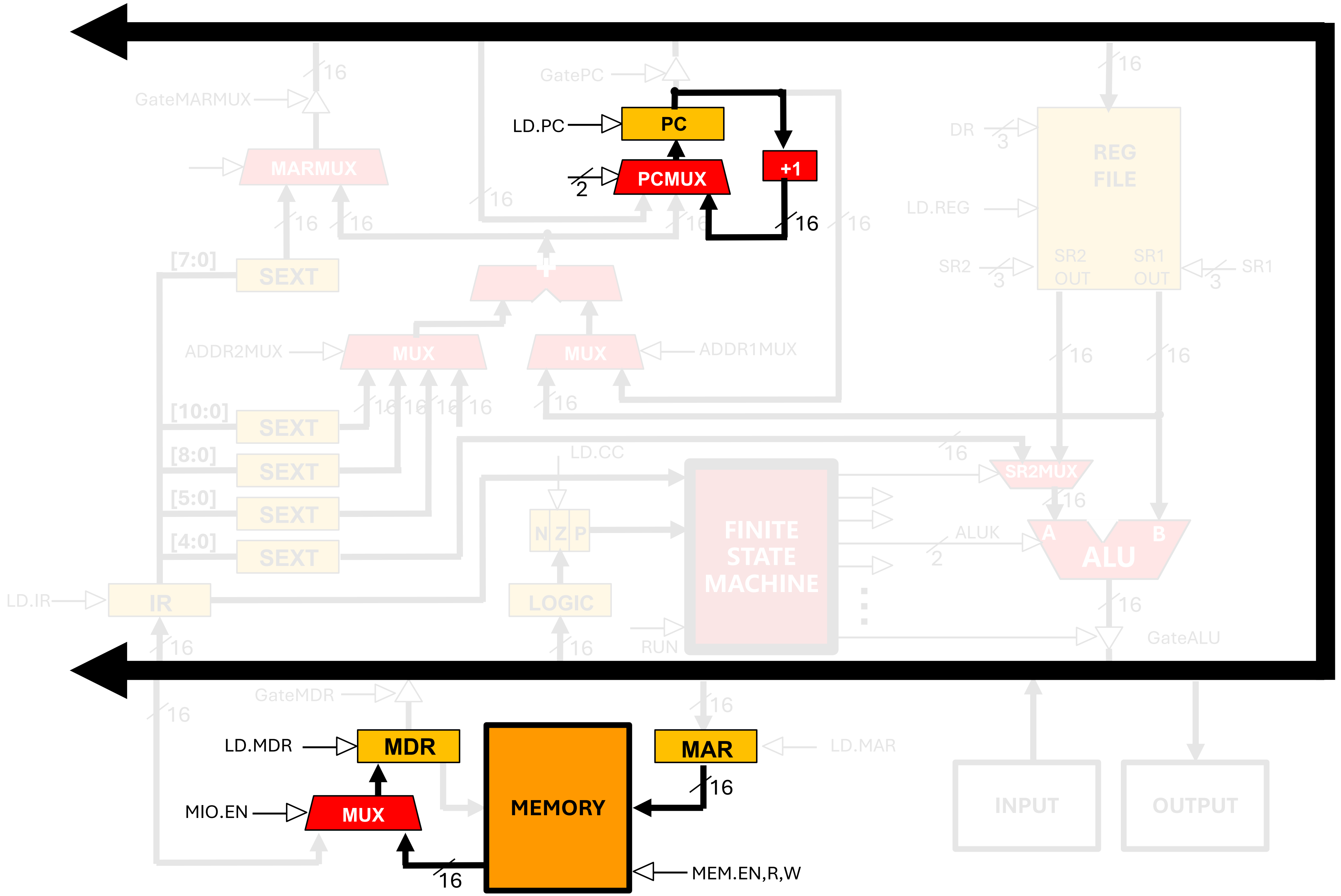
LDR (Base+Offset)



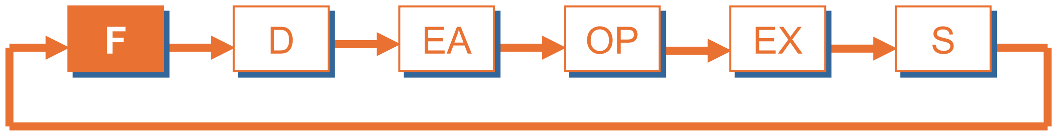


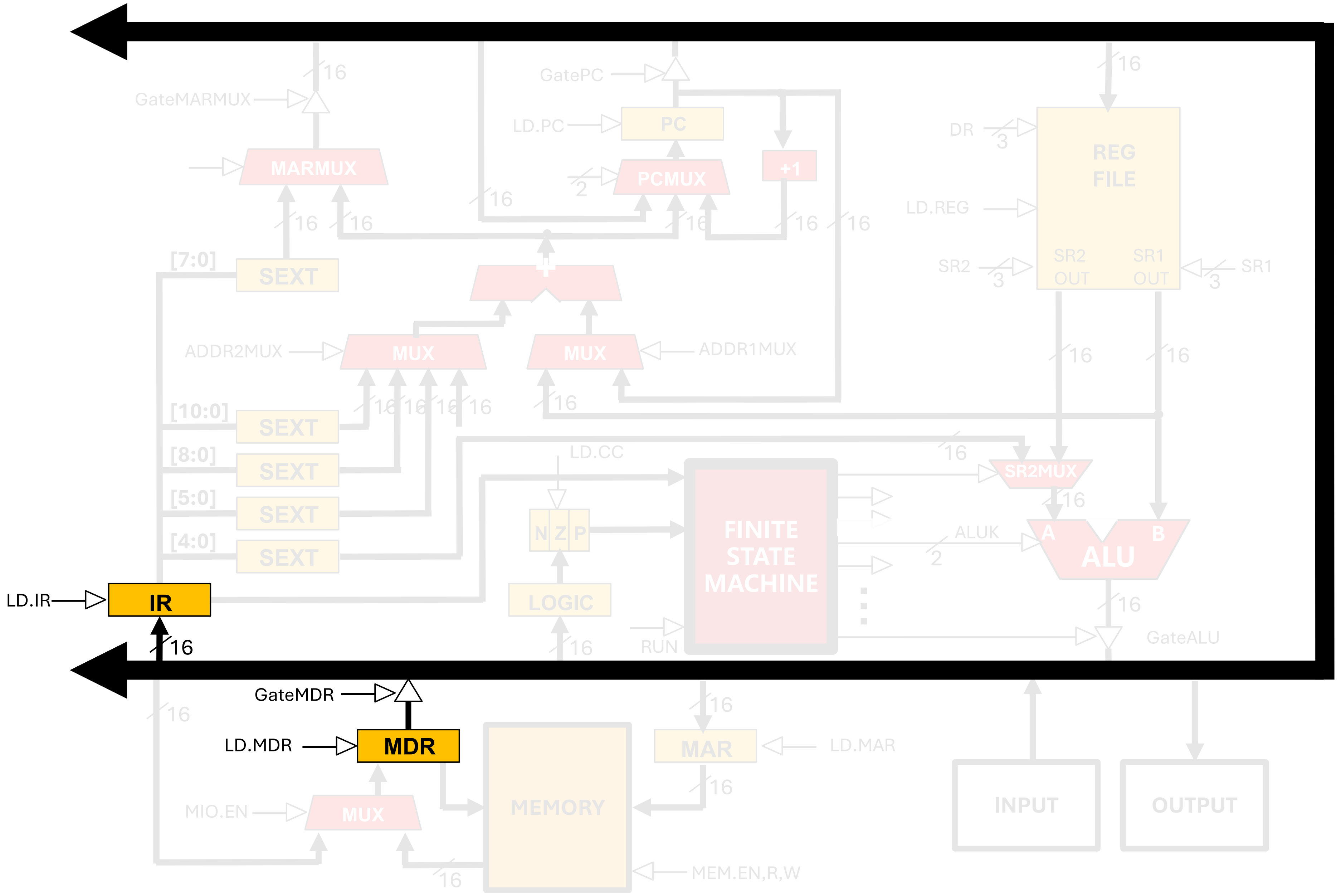
LDR (Base+Offset)



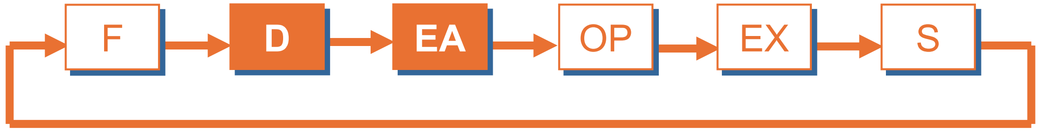


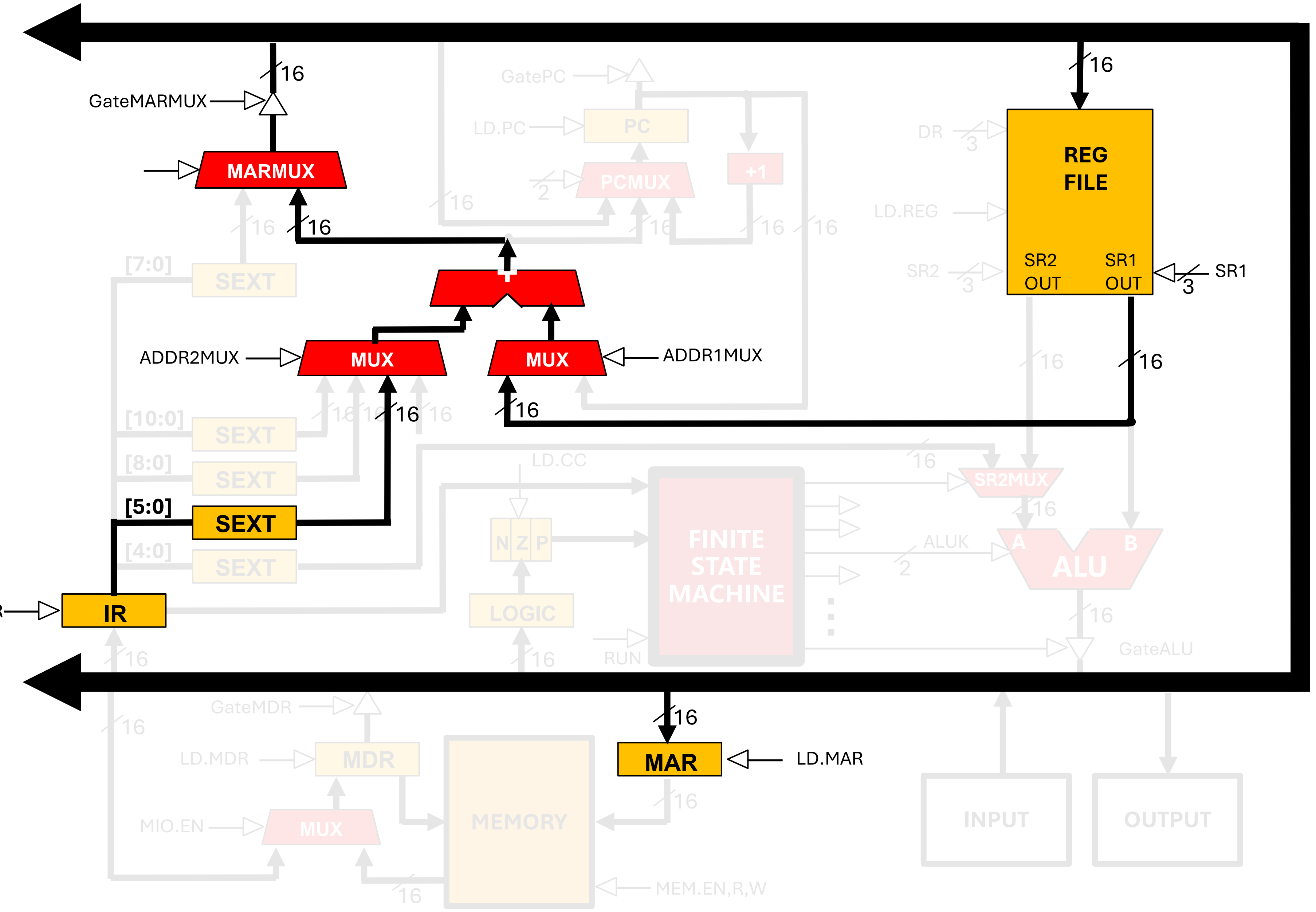
LDR (Base+Offset)



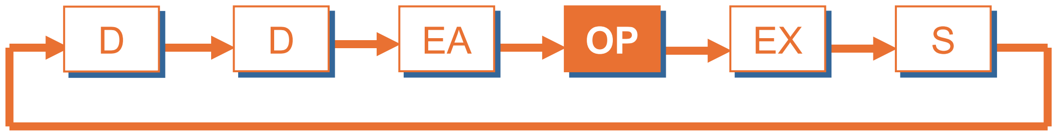


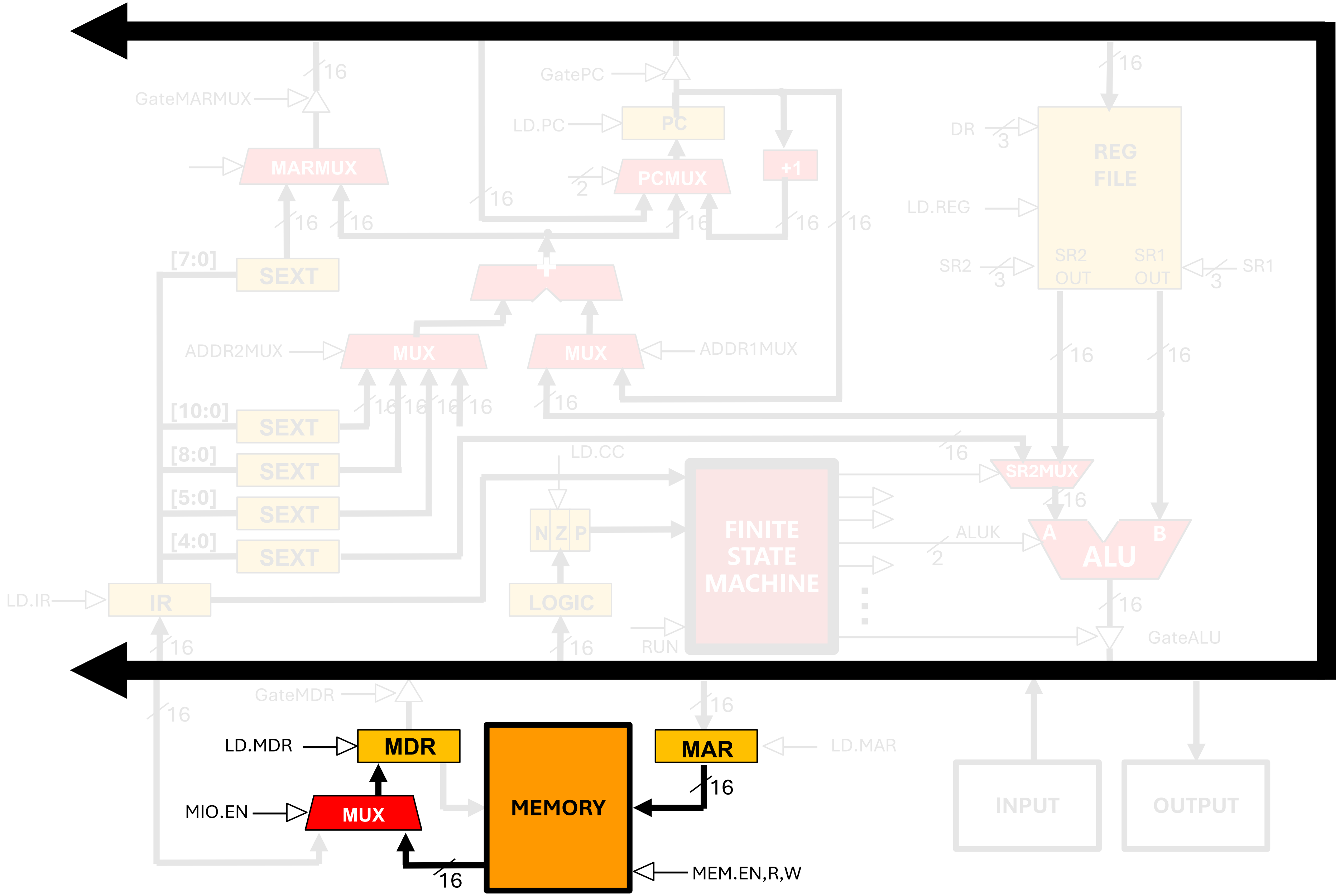
LDR (Base+Offset)



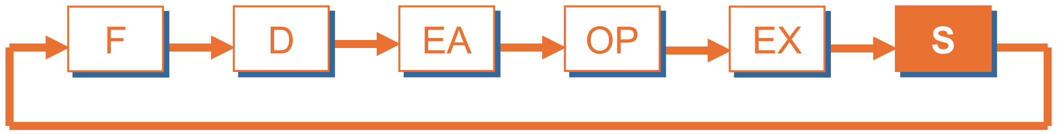


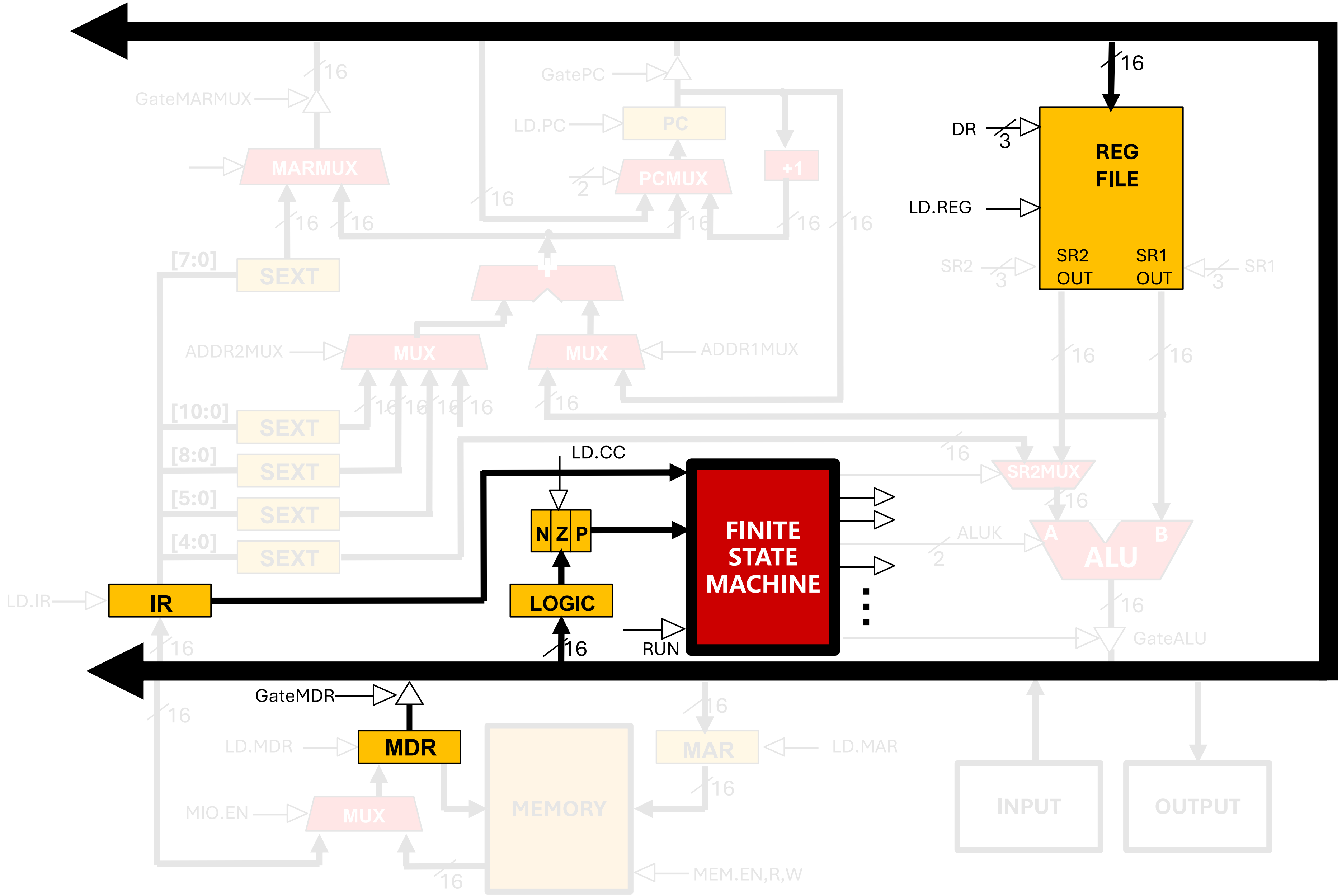
LDR (Base+Offset)





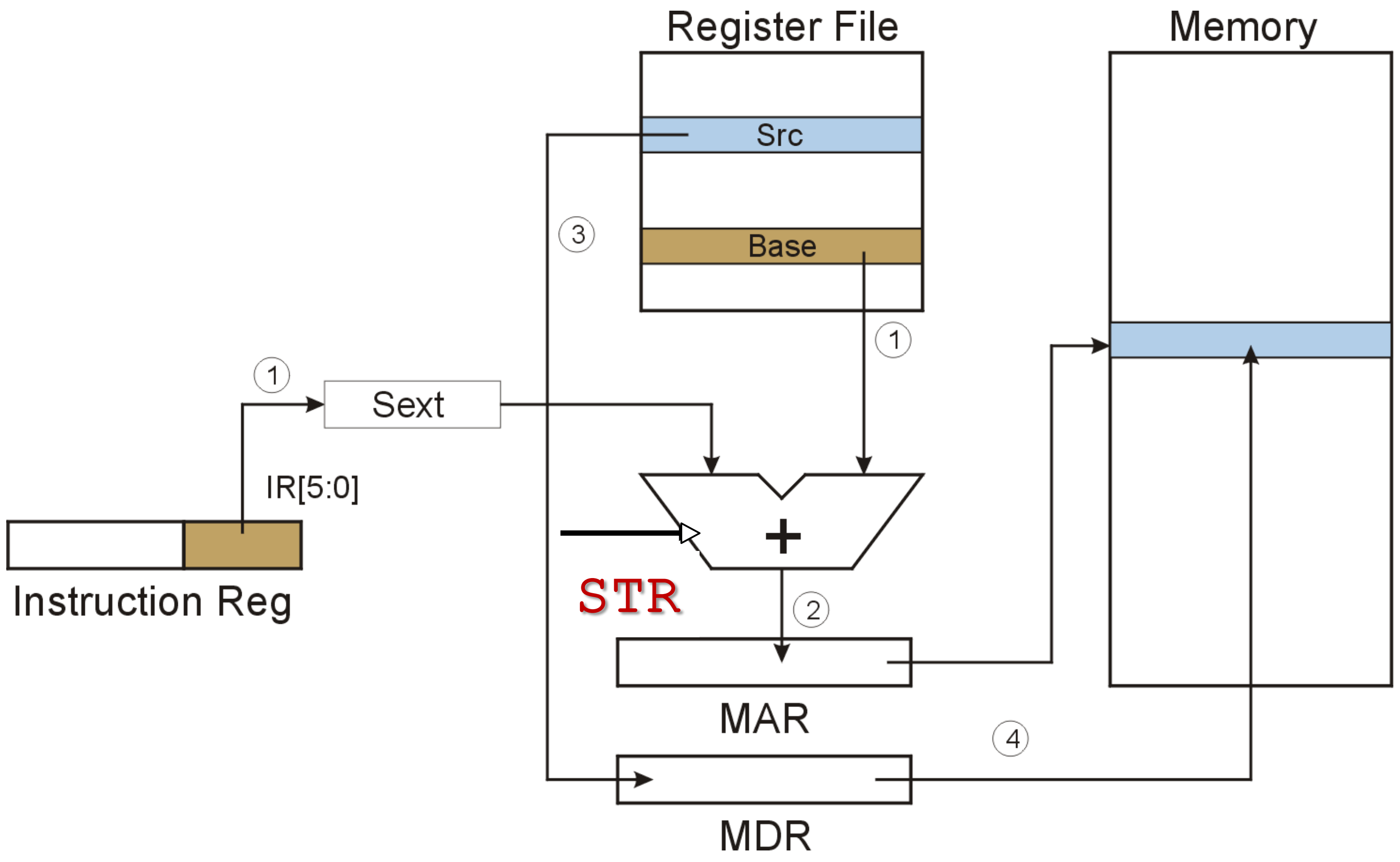
LDR (Base+Offset)



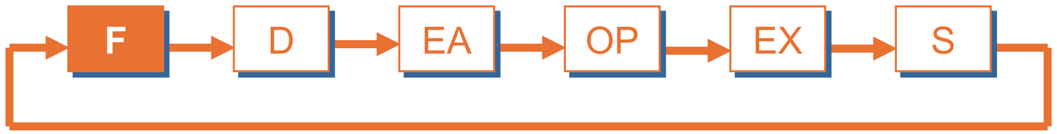


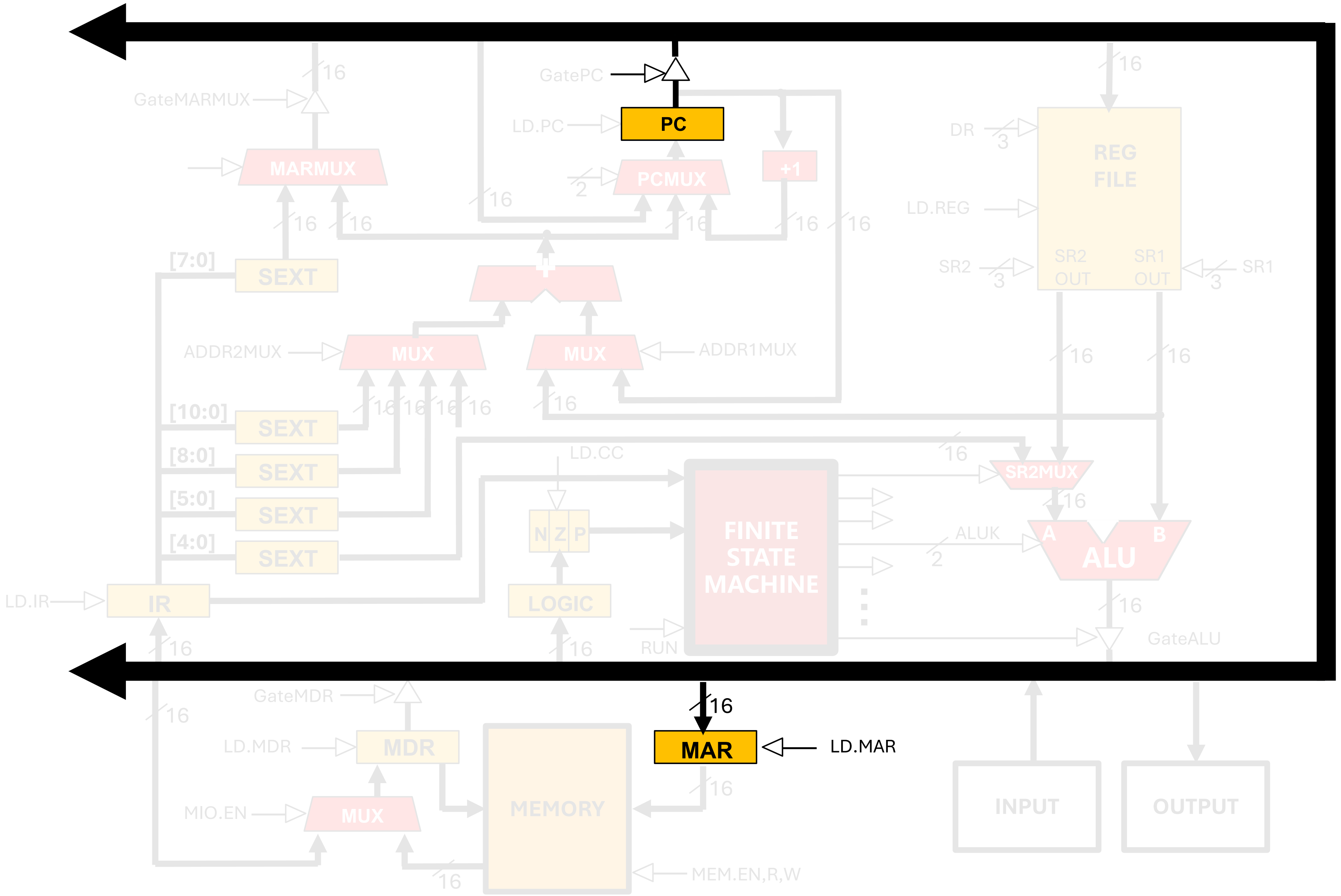
STR (Base+Offset) STR SR, BaseR, offset6

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

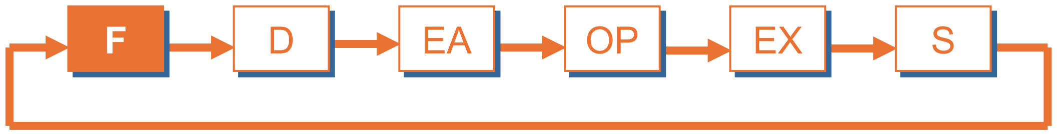


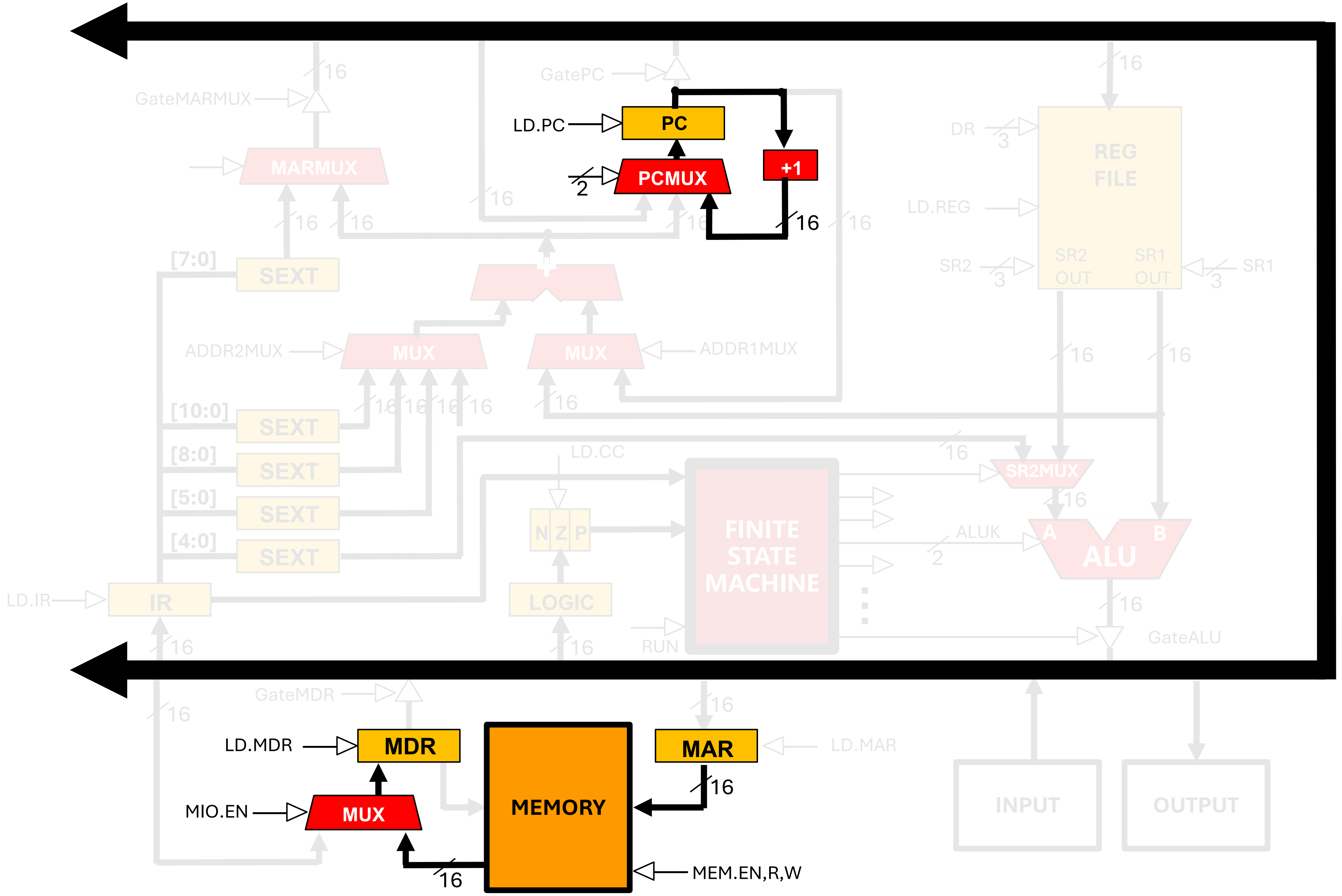
STR (Base+Offset)



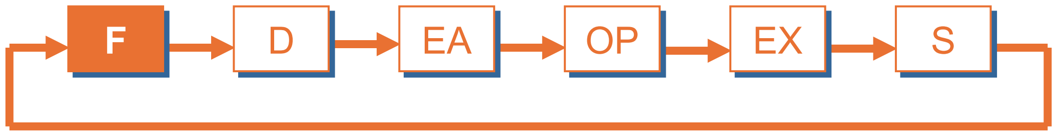


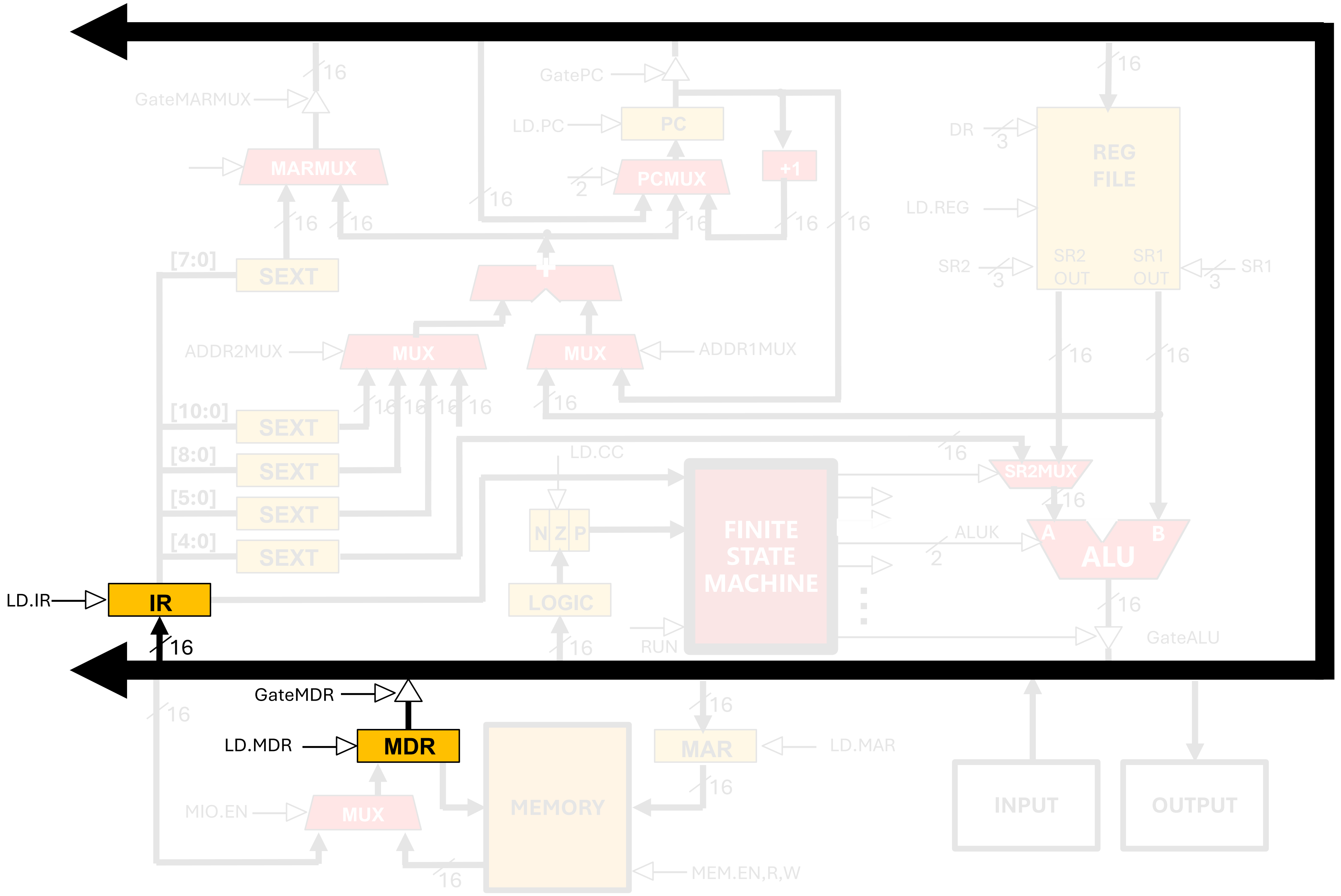
STR (Base+Offset)



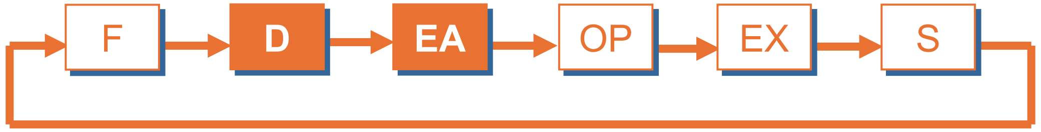


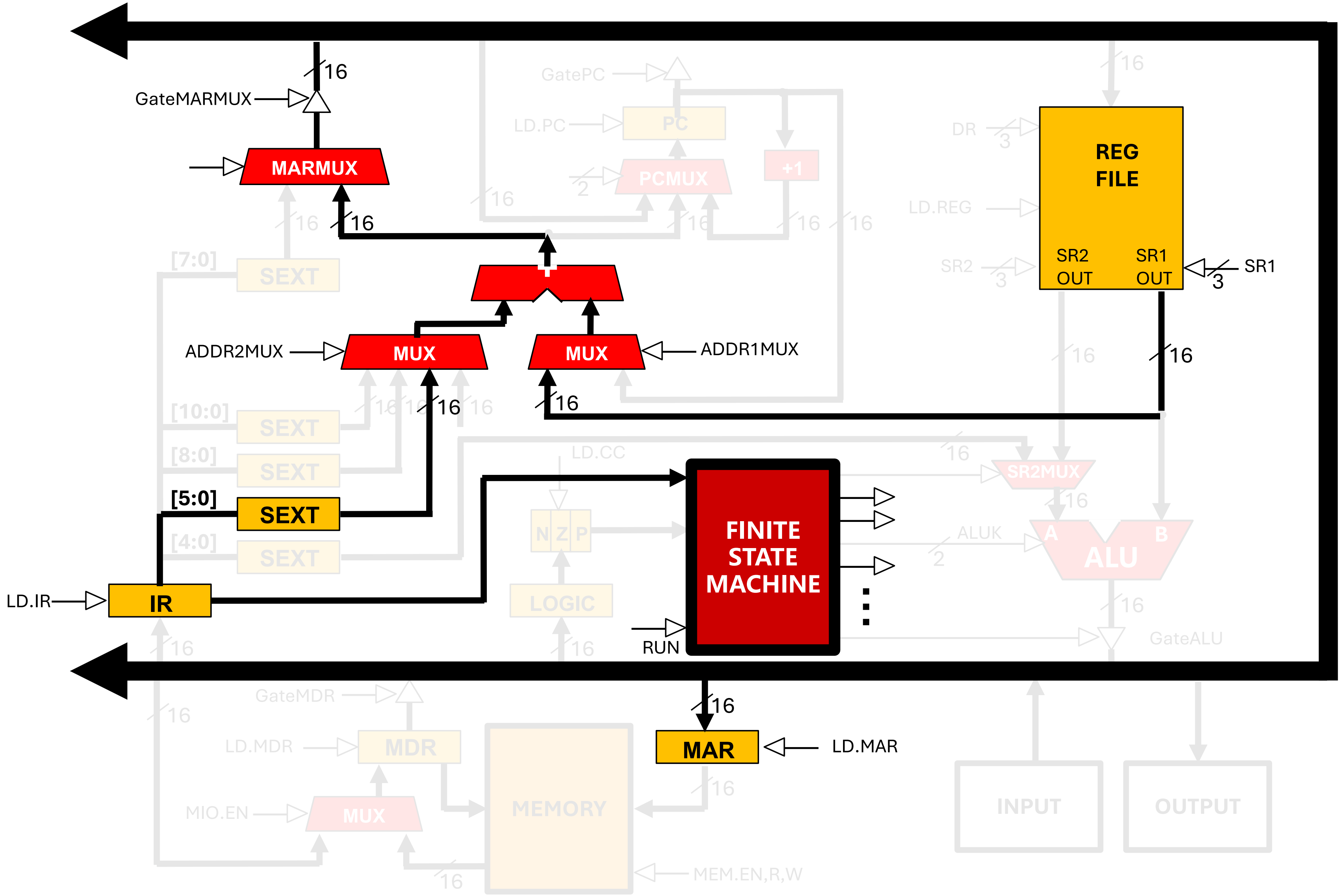
STR (Base+Offset)



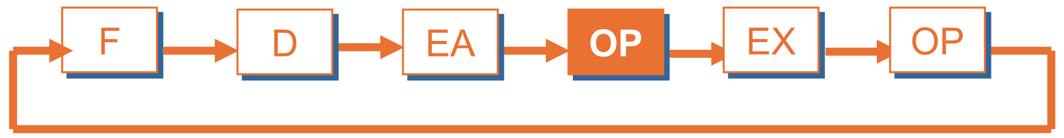


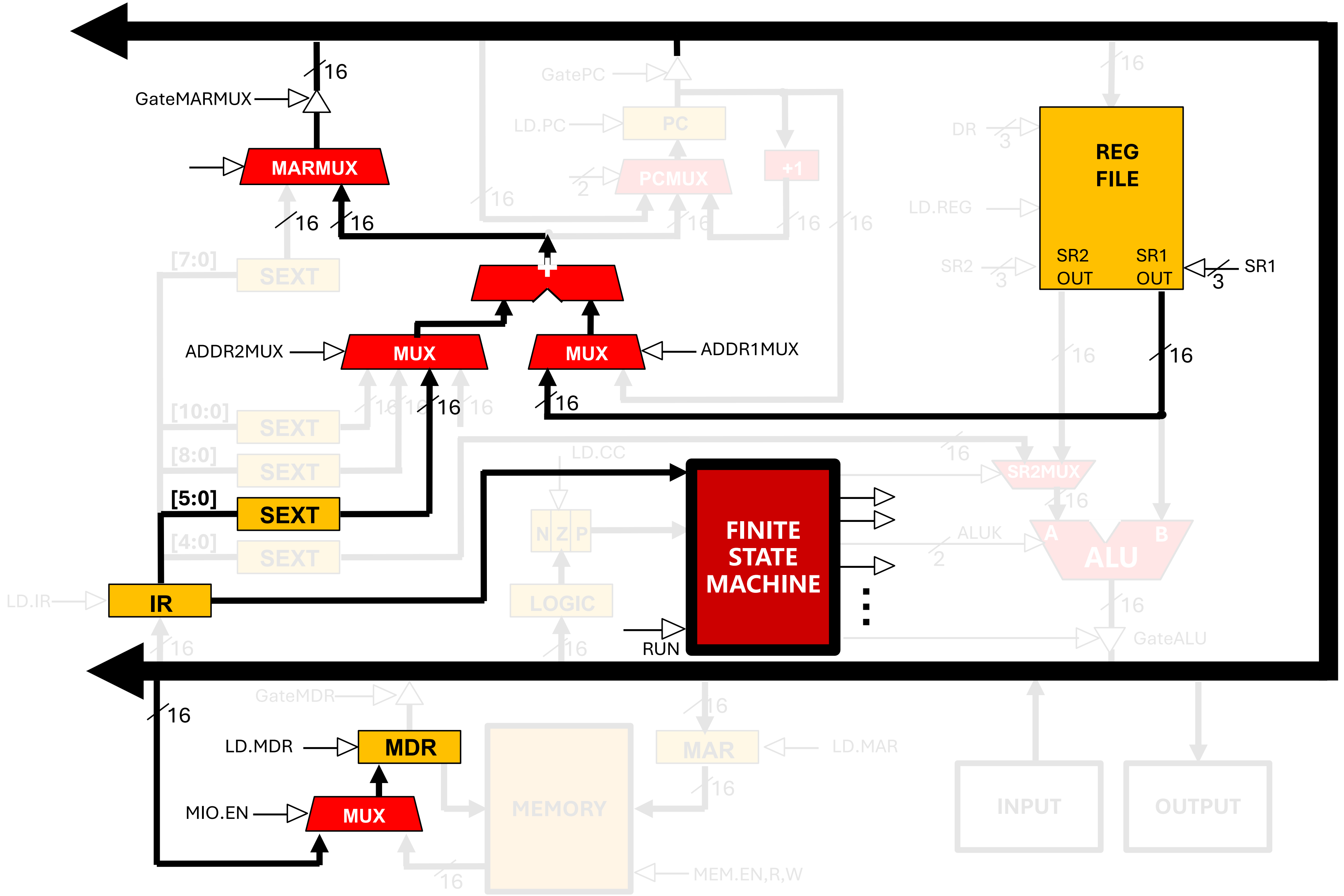
STR (Base+Offset)



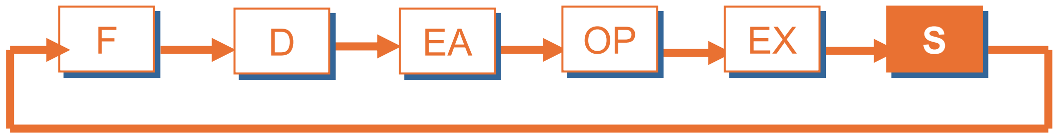


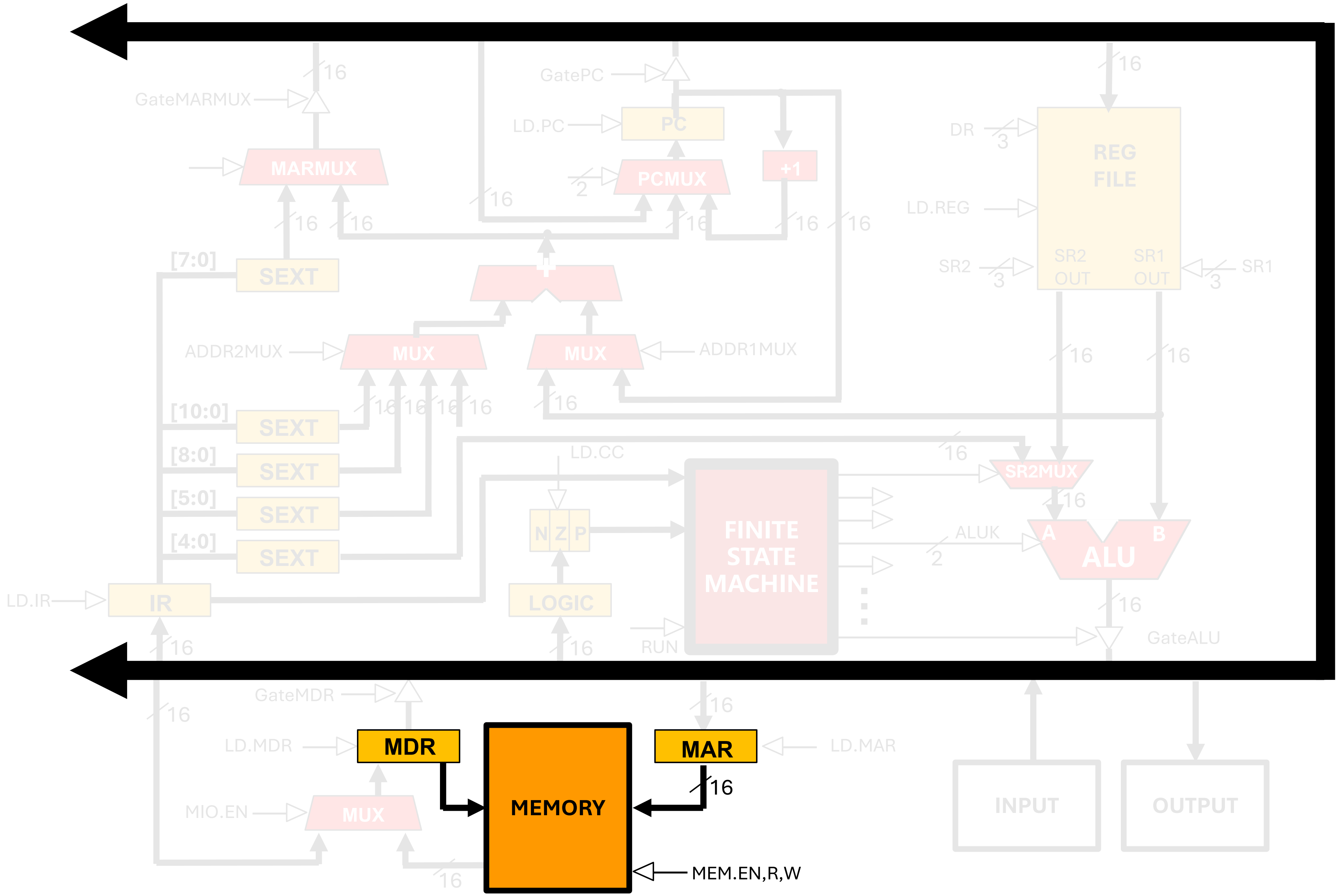
STR (Base+Offset)





STR (Base+Offset)





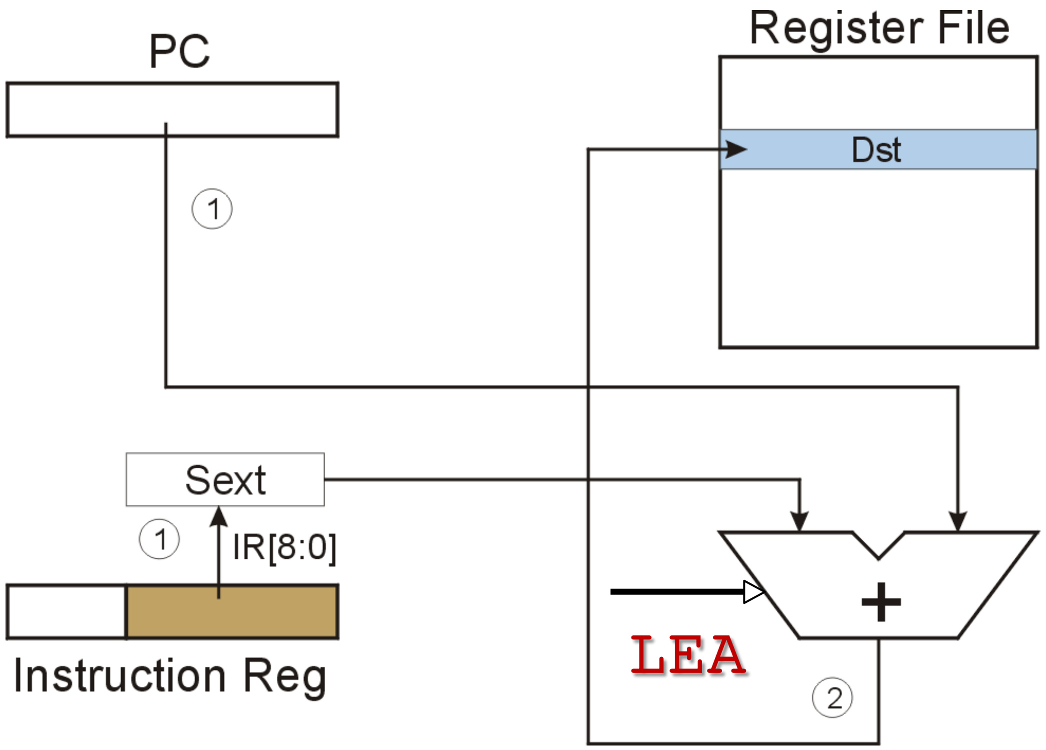
Load Effective Address

Computes address like PC-relative **(PC plus signed offset)** and **stores the result into a register**.

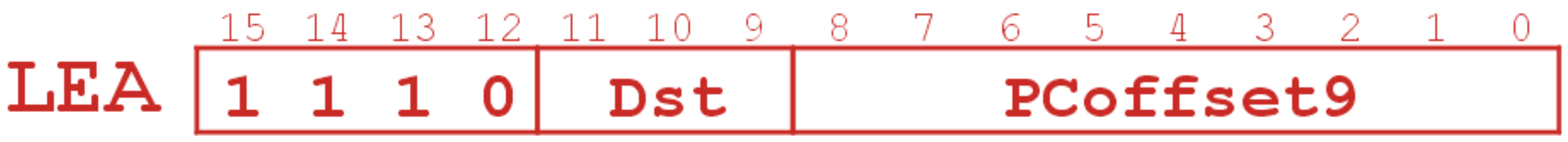
Note: The *address* is stored in the register, not the contents of the memory location.

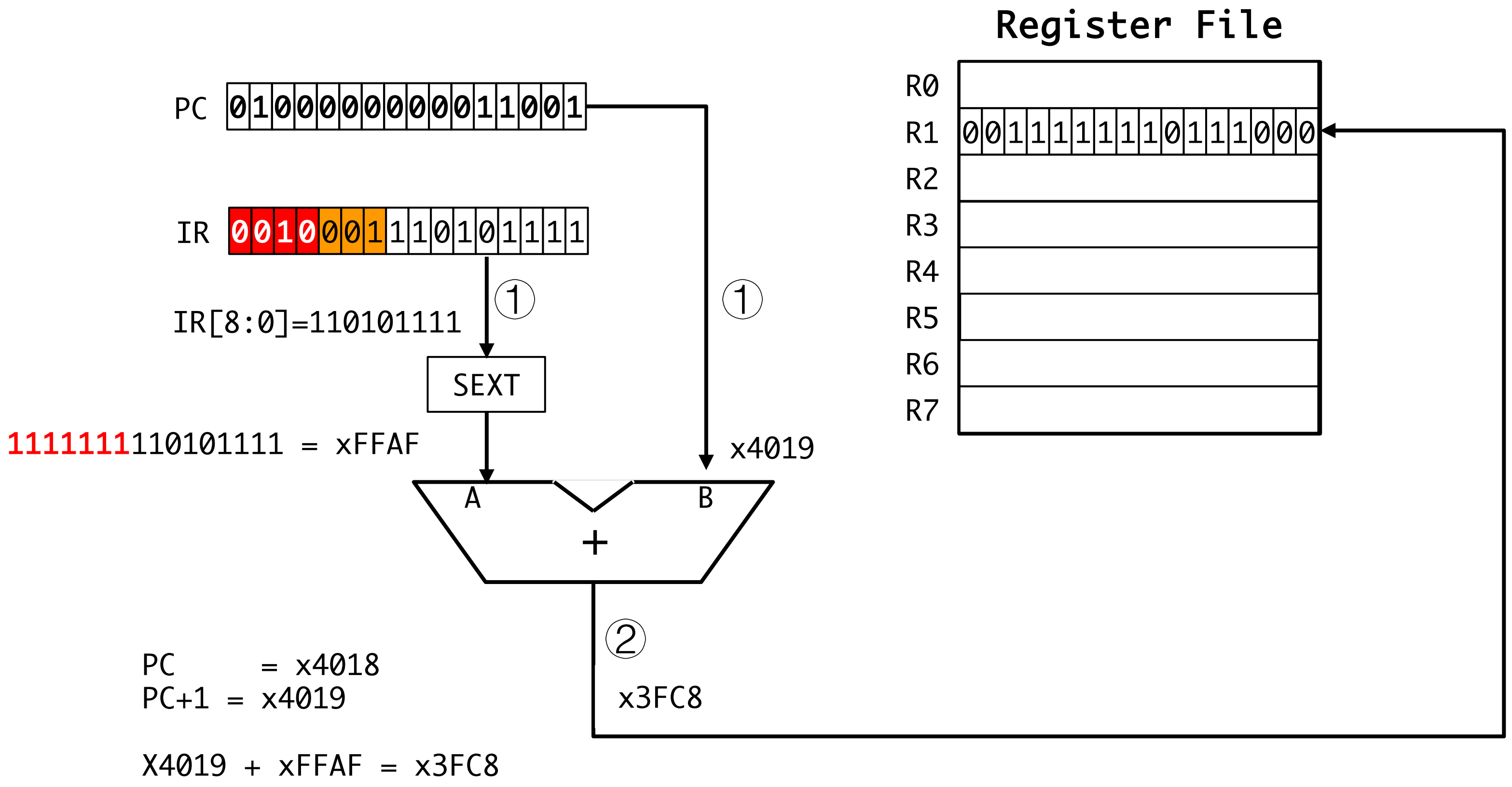
LEA (Immediate) LD DR, PCoffset9

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



LEA (Immediate): LEA R1, x1AF

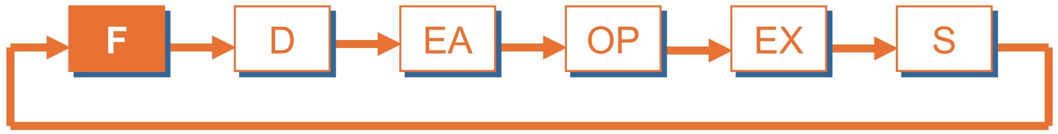


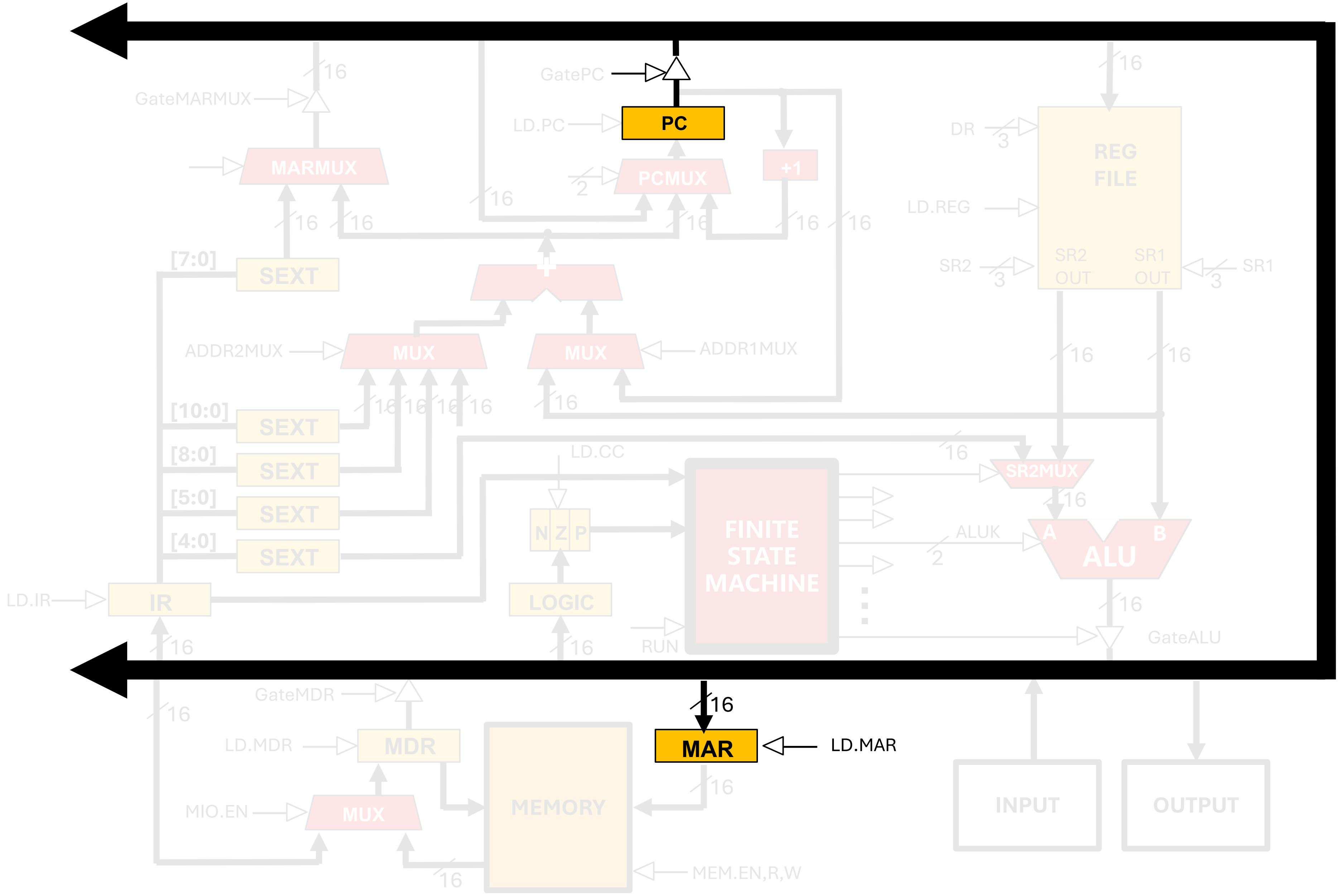


PC = x4018
PC+1 = x4019

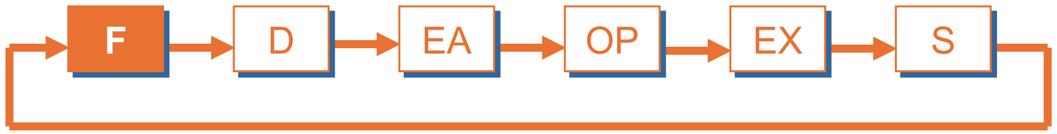
X4019 + xFFAF = x3FC8

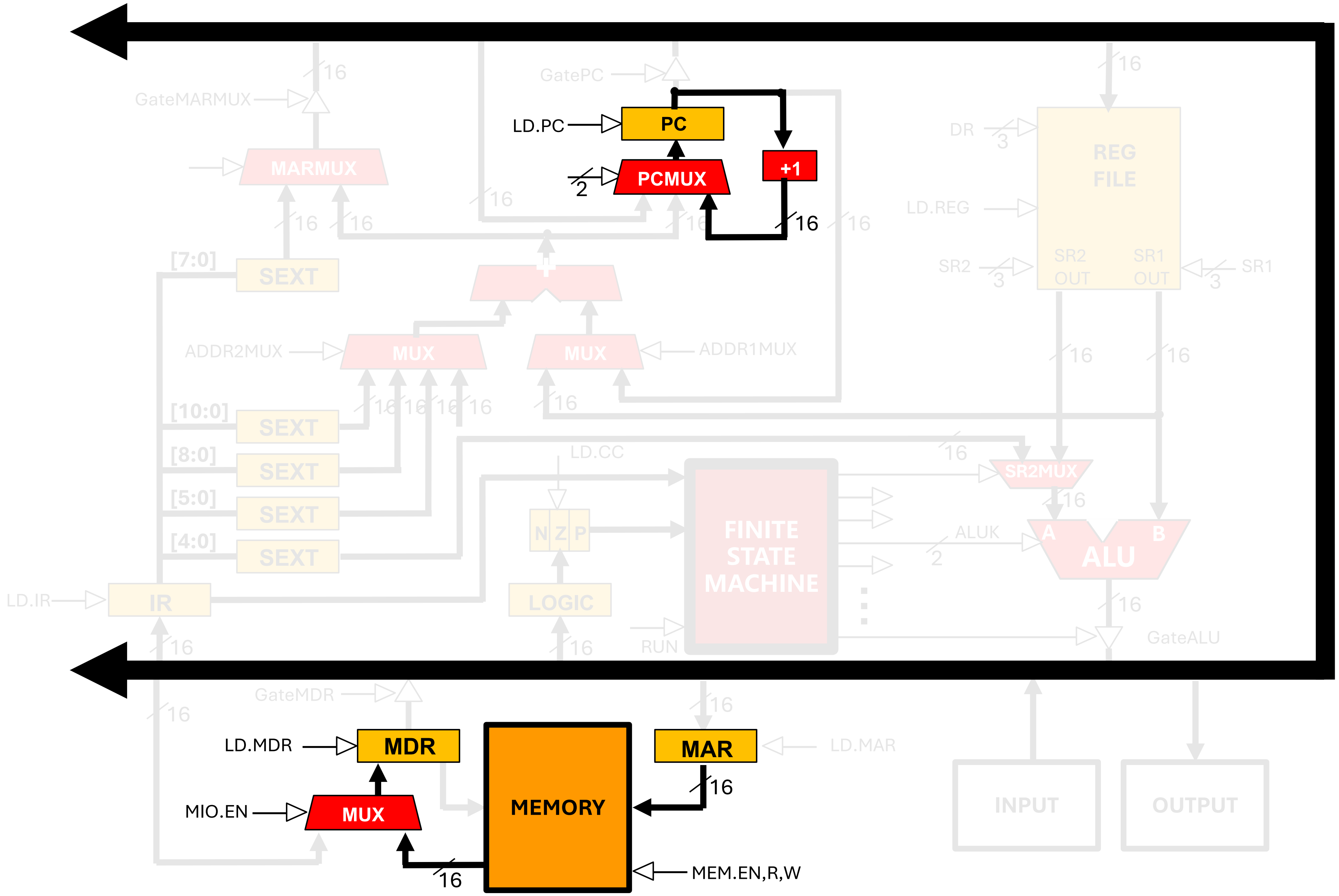
LEA (Immediate)



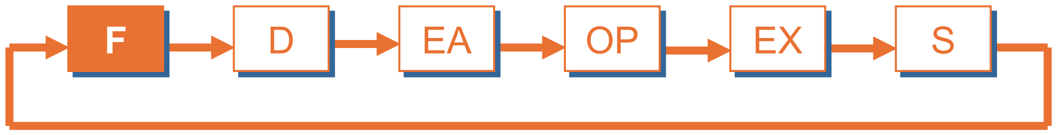


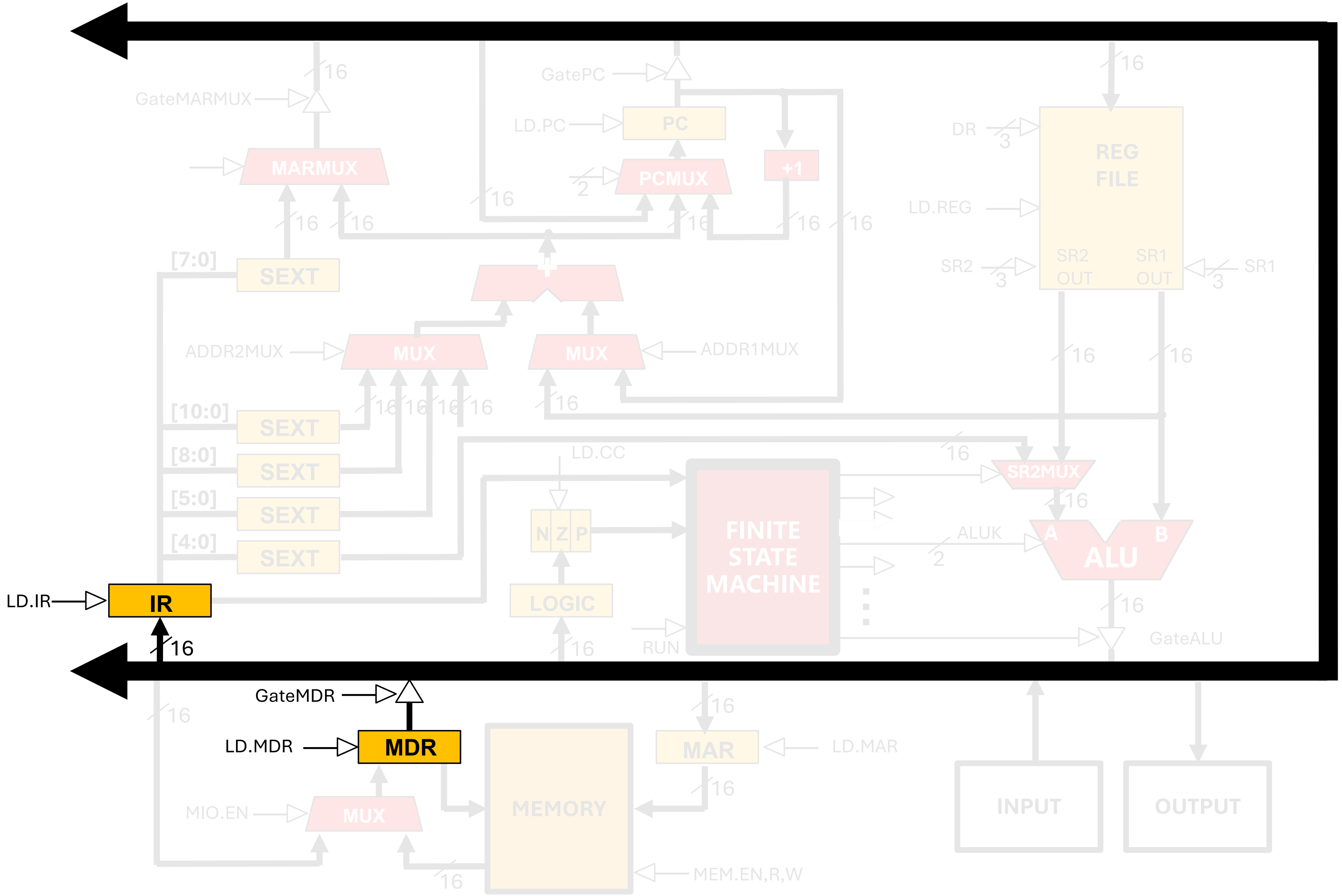
LEA (Immediate)



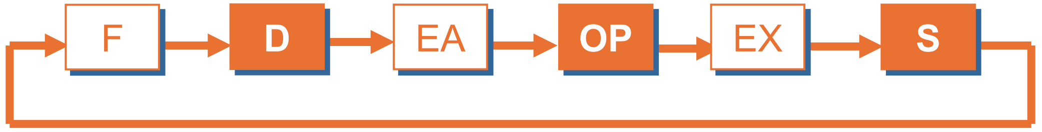


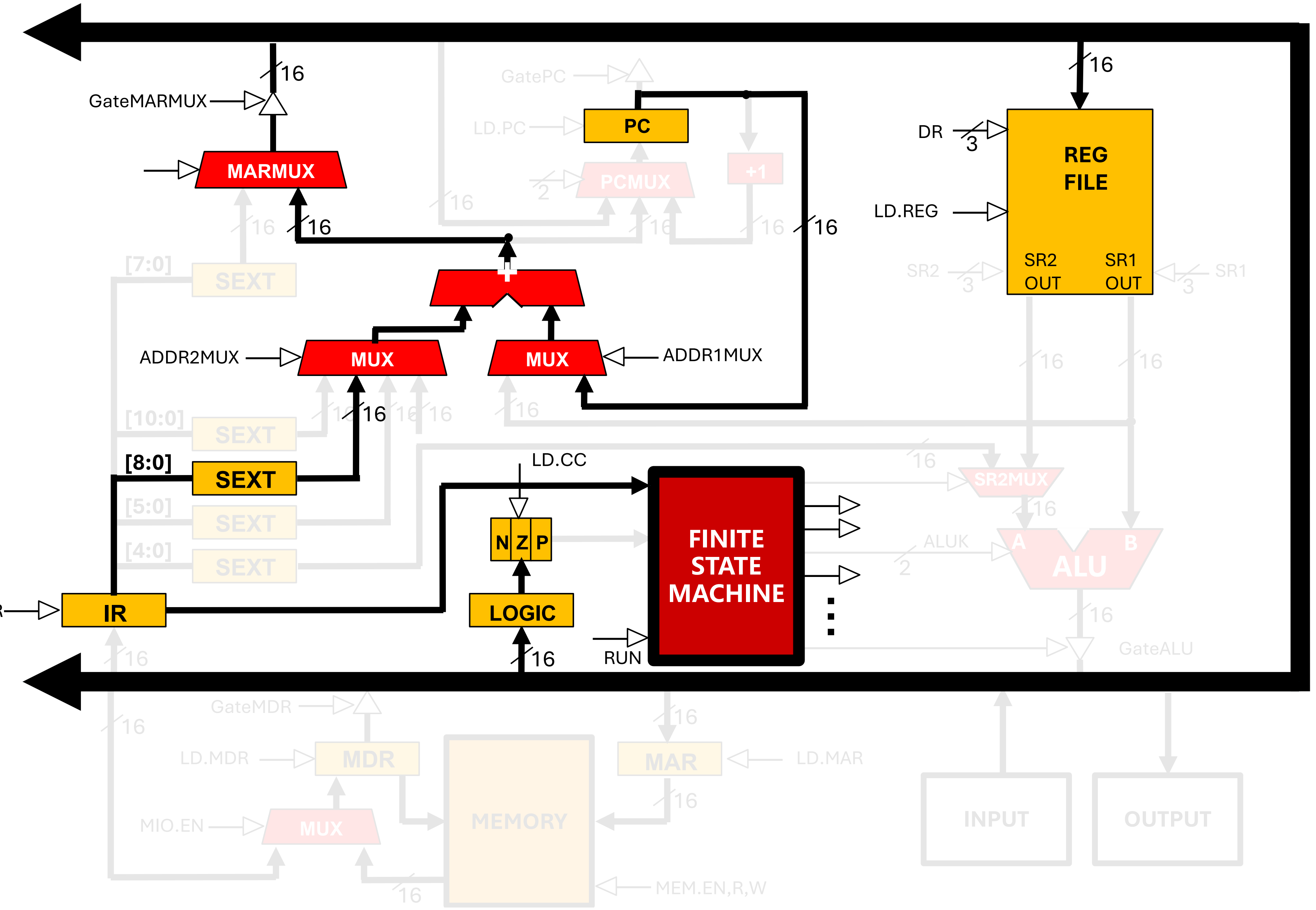
LEA (Immediate)





LEA (Immediate)





Example

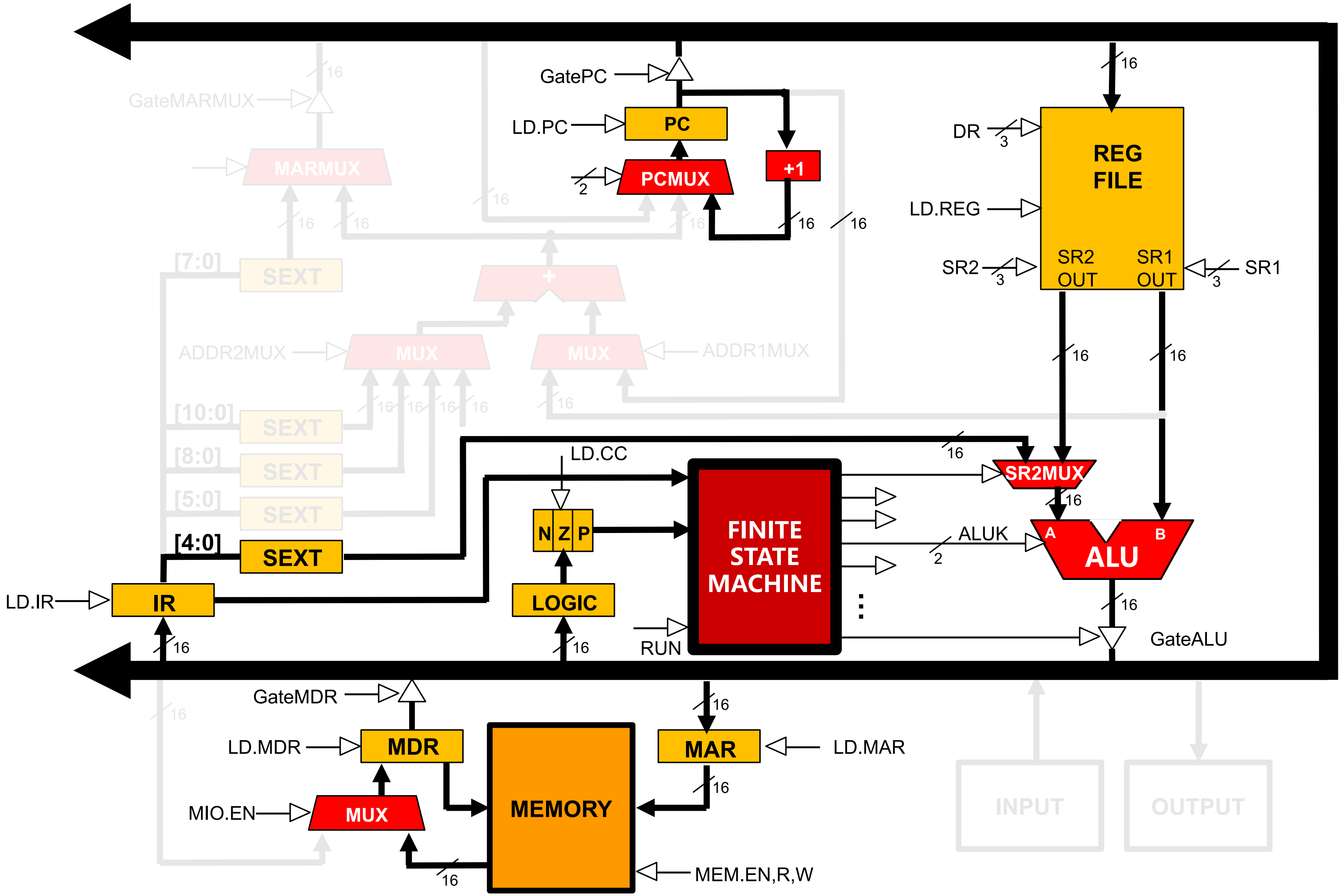
0001	ADD
0011	ST
0101	AND
0111	STR
1010	LDI
1110	LEA

Address	Instruction	Comments
x30F6	1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1	R1 ← PC – 3 = x30F4
x30F7	0 0 0 1 0 1 0 0 0 1 1 0 1 1 1 0	R2 ← R1 + 14 = x3102
x30F8	0 0 1 1 0 1 0 1 1 1 1 1 1 0 1 1	M[PC - 5] ← R2; i.e. M[x30F4] ← x3102
x30F9	0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0	R2 ← 0
x30FA	0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1	R2 ← R2 + 5 = 5
x30FB	0 1 1 1 0 1 0 0 0 1 0 0 1 1 1 0	M[R1+14] ← R2; i.e. M[x3102] ← 5
x30FC	1 0 1 0 0 1 1 1 1 1 1 1 0 1 1 1	R3 ← M[M[PC-9]] = M[M[x30F4]] = M[x3102] = 5

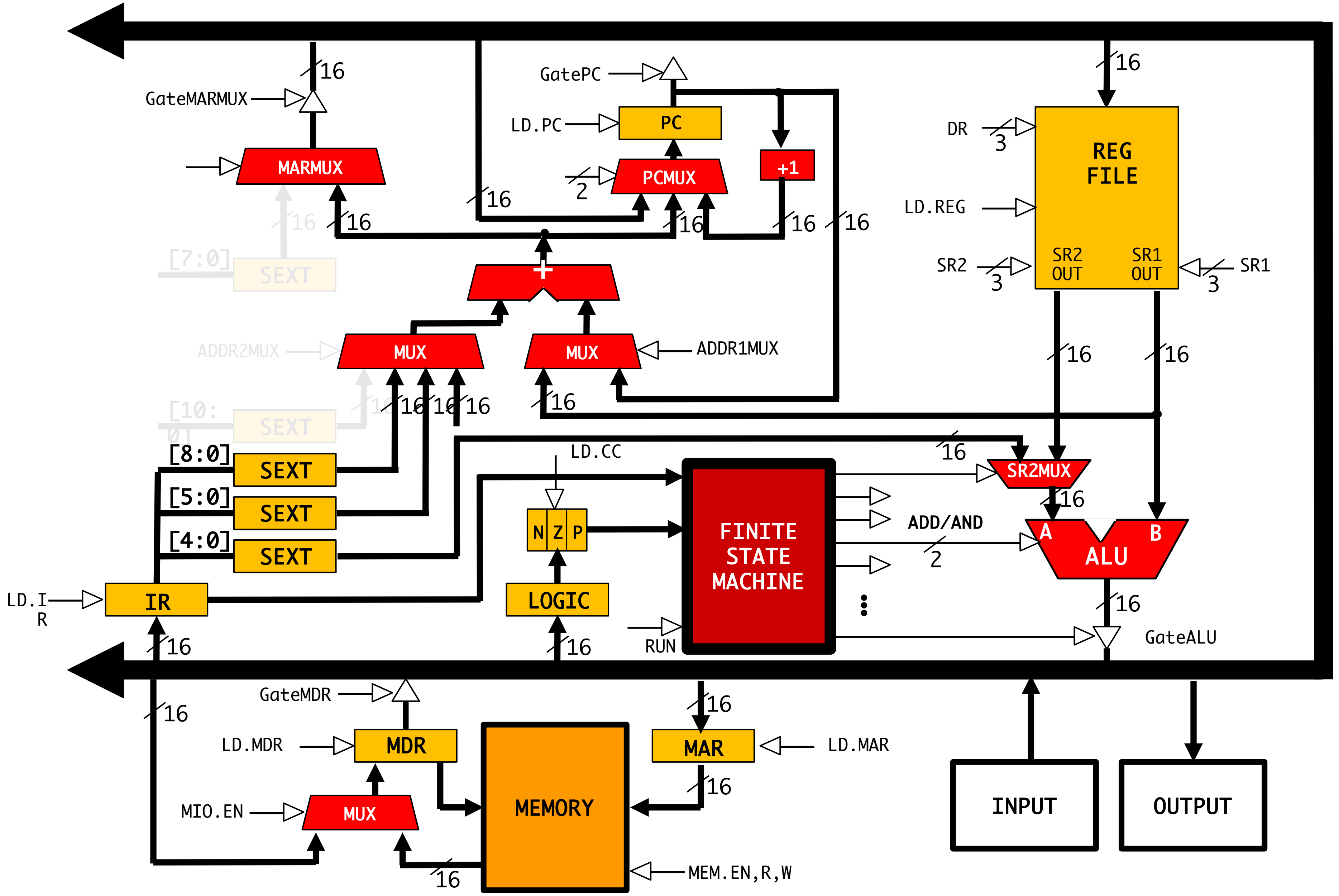
opcode

Summary

LC-3 Data Path After *Operate Instruction*



LC-3 Data Path After *Load/Store Instruction*



Q & A