Control Instructions

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR	0	0	0	0	n	z	p	PCoffset9								
JSR	0	1	0	0	1	PCoffset11										
JSRR	0	1	0	0	0	0	0	BaseR			0	0	0	0	0	0
RTI	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JMP	1	1	0	0	0	0	0	BaseR			0	0	0	0	0	0
RET	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
TRAP	1	1	1	1	0	0	0	0	TrapVector8							

Conditional Branch Instruction

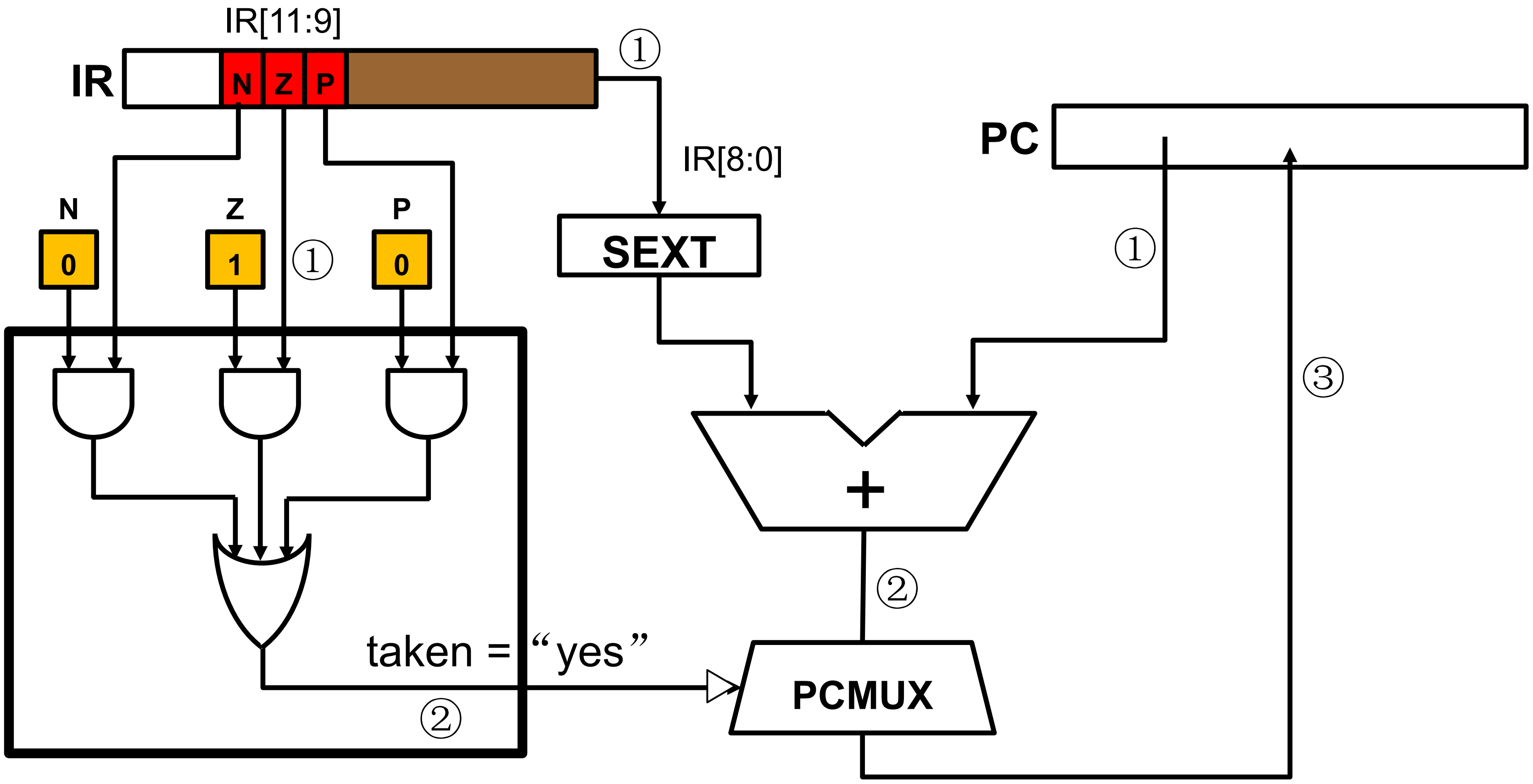
Branch specifies one or more condition codes.
If the specified bit is set, the branch is taken.

- **PC-relative addressing:**
  target address is made by adding signed offset (IR[8:0]) to current PC.
- **Note: PC has already been incremented by FETCH stage.**
- **Note: Target must be within 256 words of BR instruction.**

If the branch is not taken, the next sequential instruction is executed.

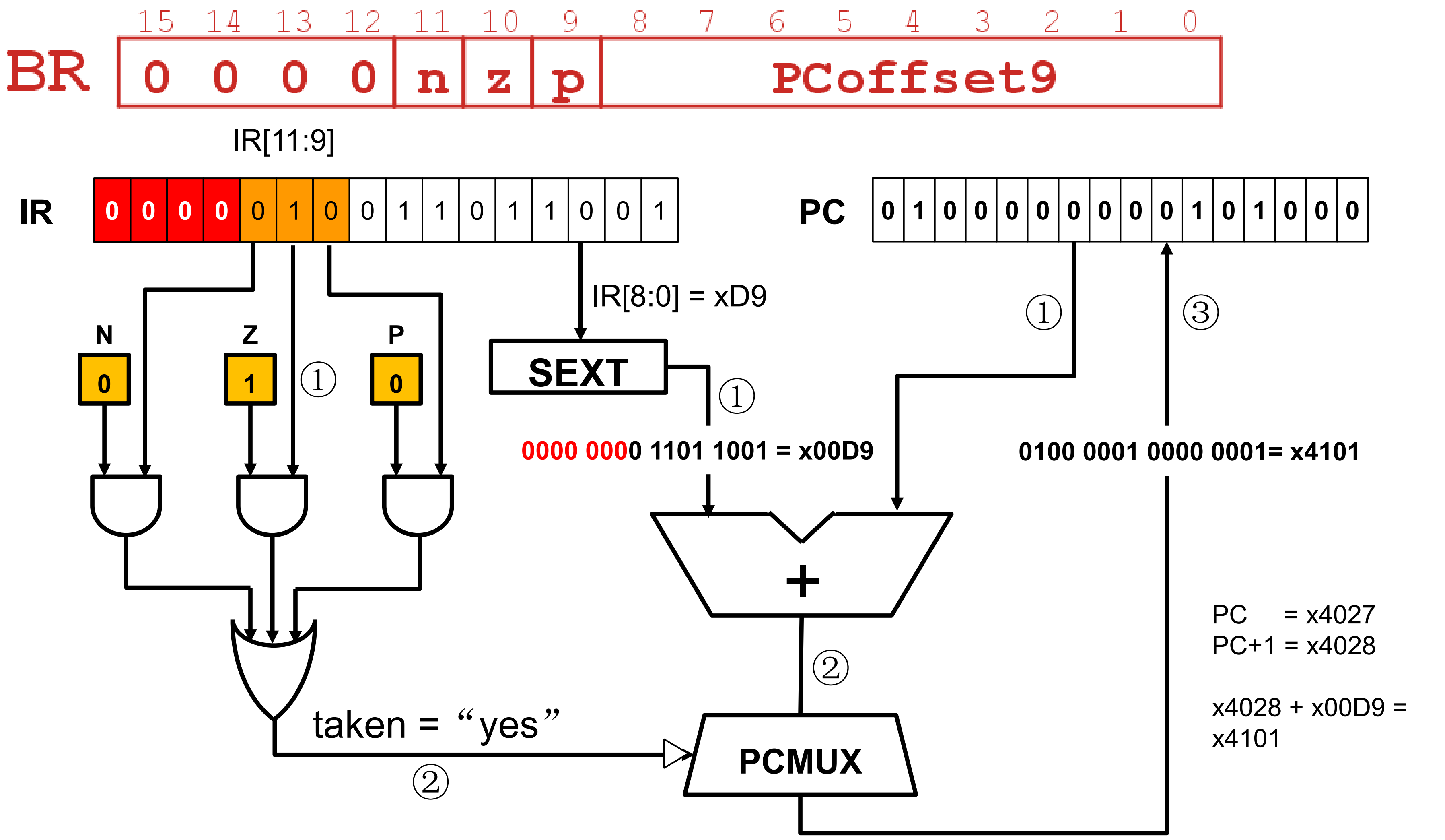
BR (PC-Relative)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR	0	0	0	0	n	z	p	PCoffset9								



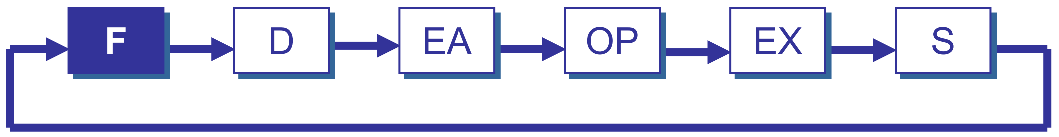
What happens if bits [11:9] are all zero?
What happens if bits [11:9] are all one?

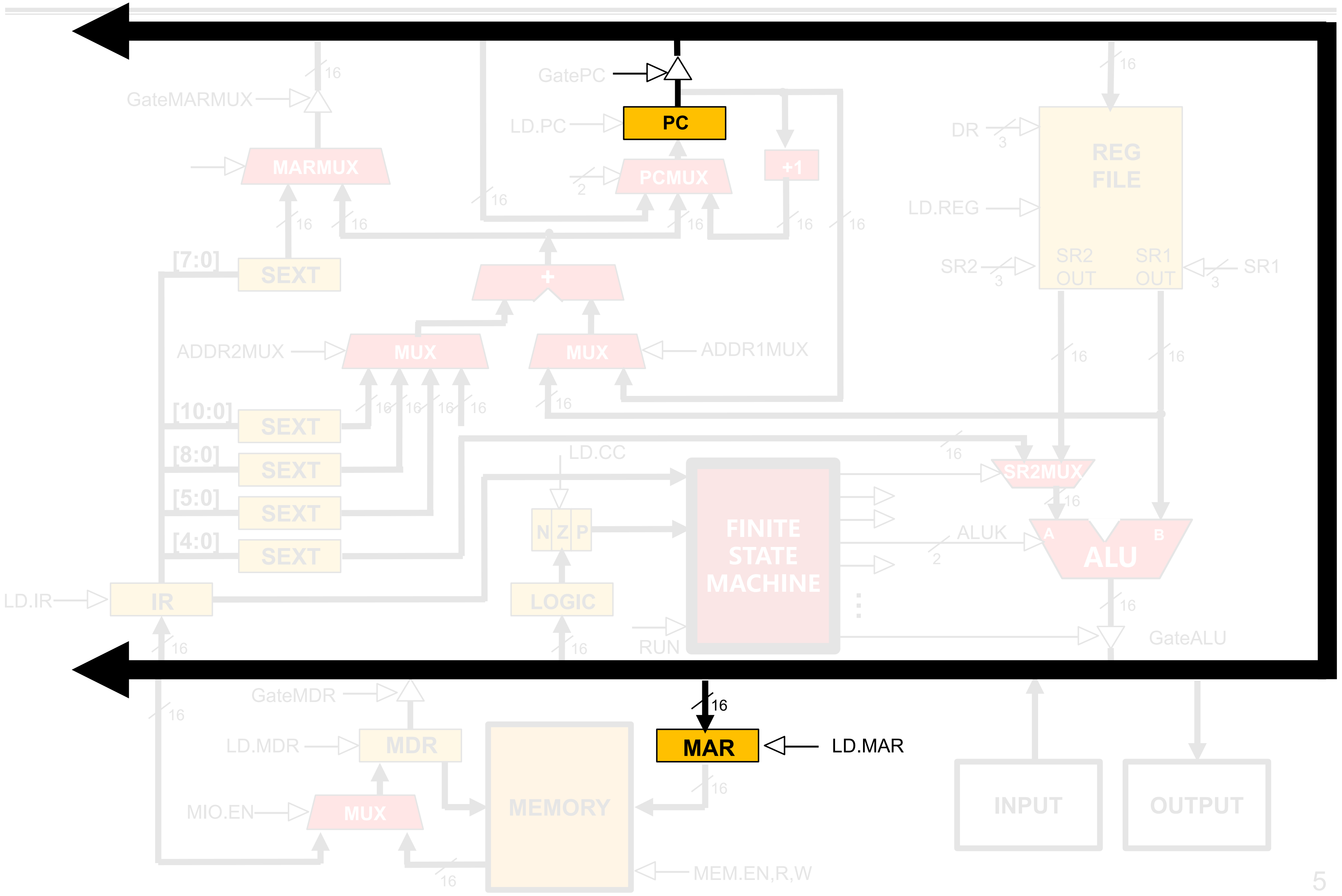
BR (PC-Relative): BR_z x4101



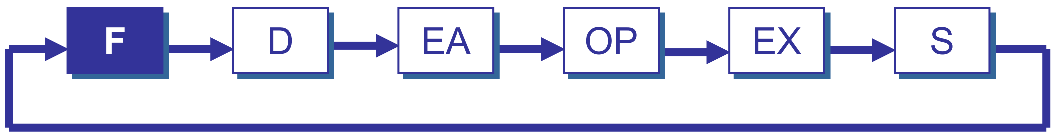
What happens if bits [11:9] are all zero?
What happens if bits [11:9] are all one?

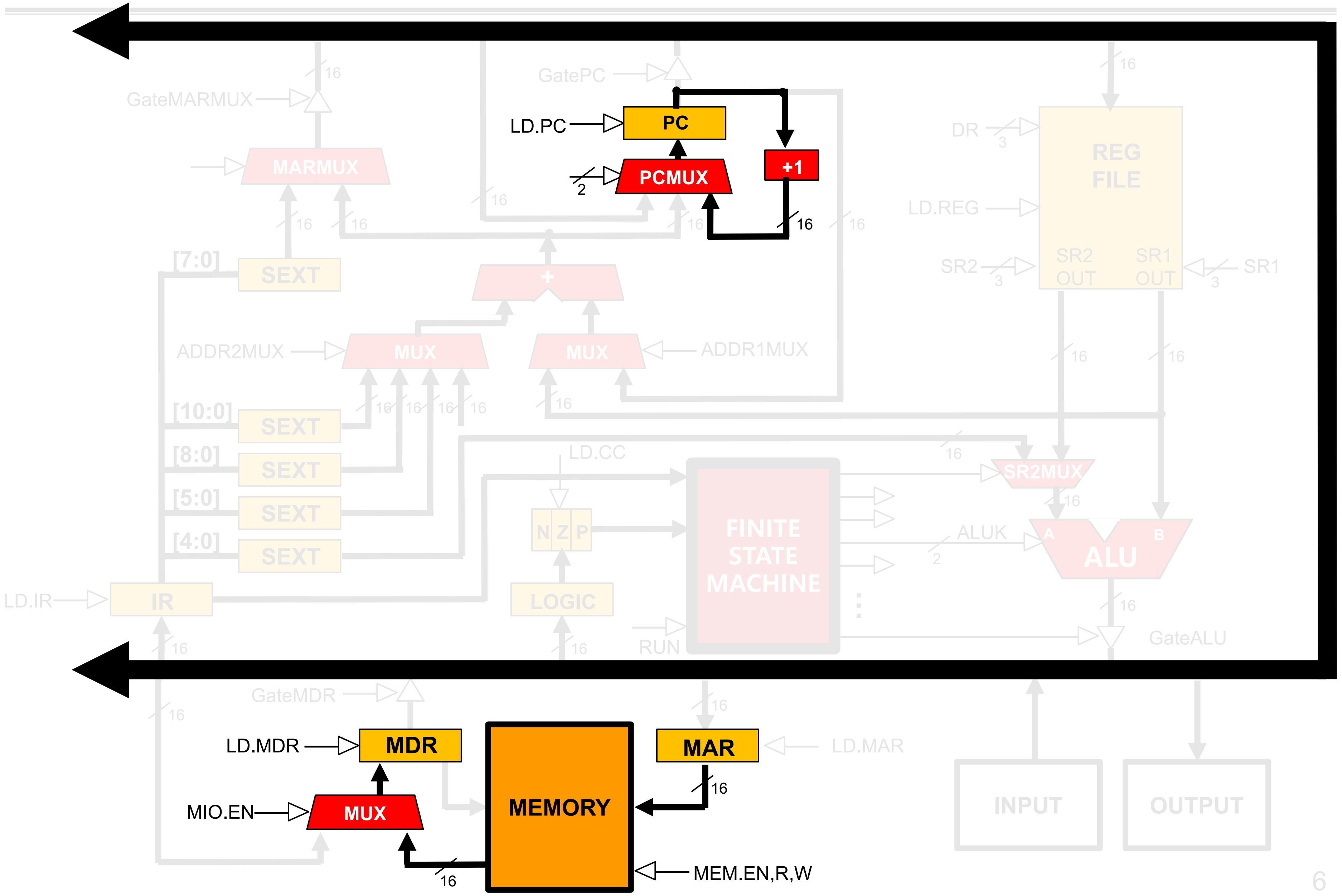
BR (PC-Relative)



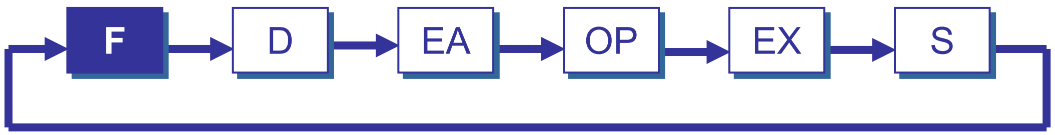


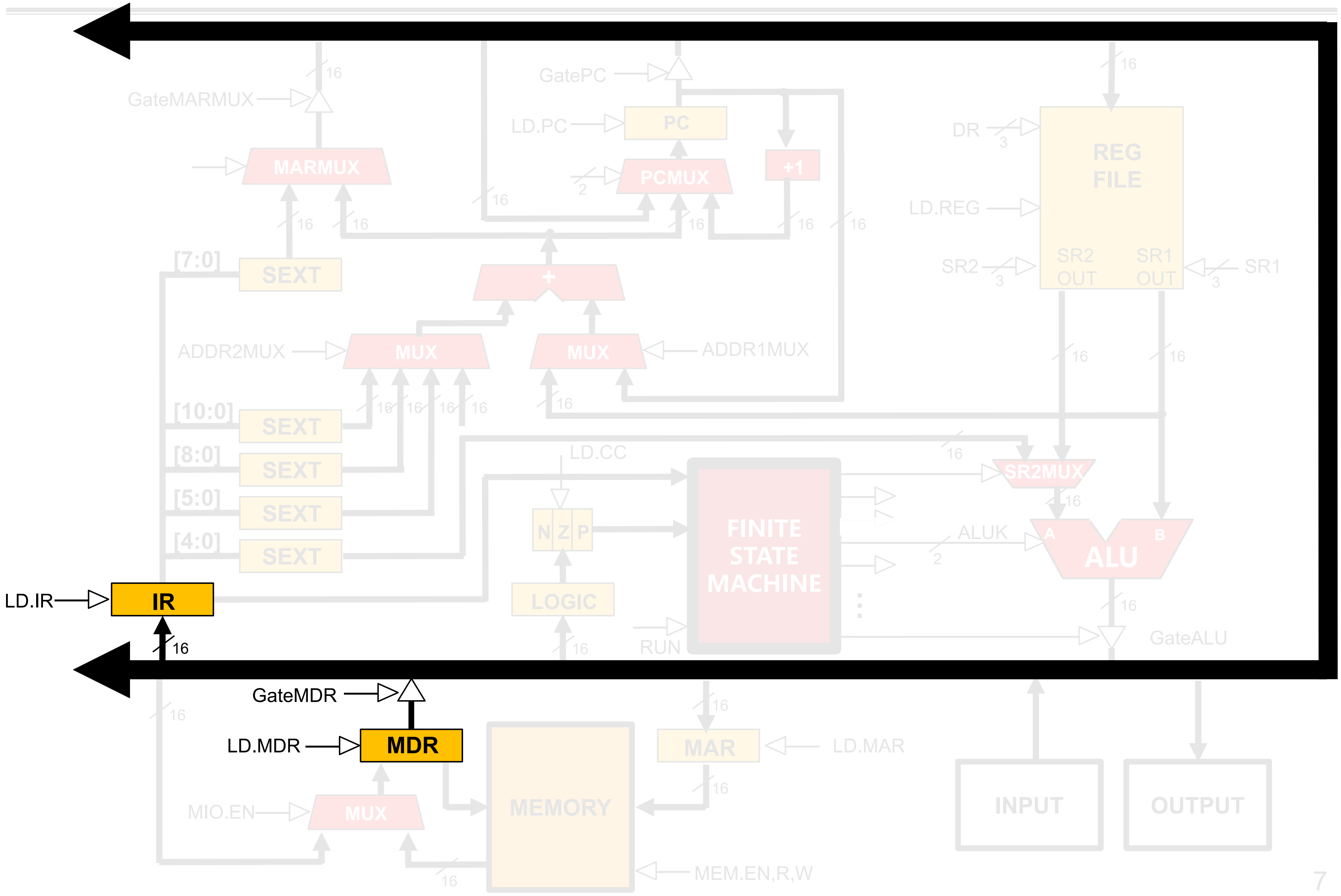
BR (PC-Relative)



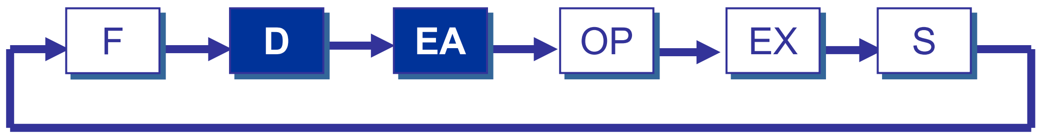


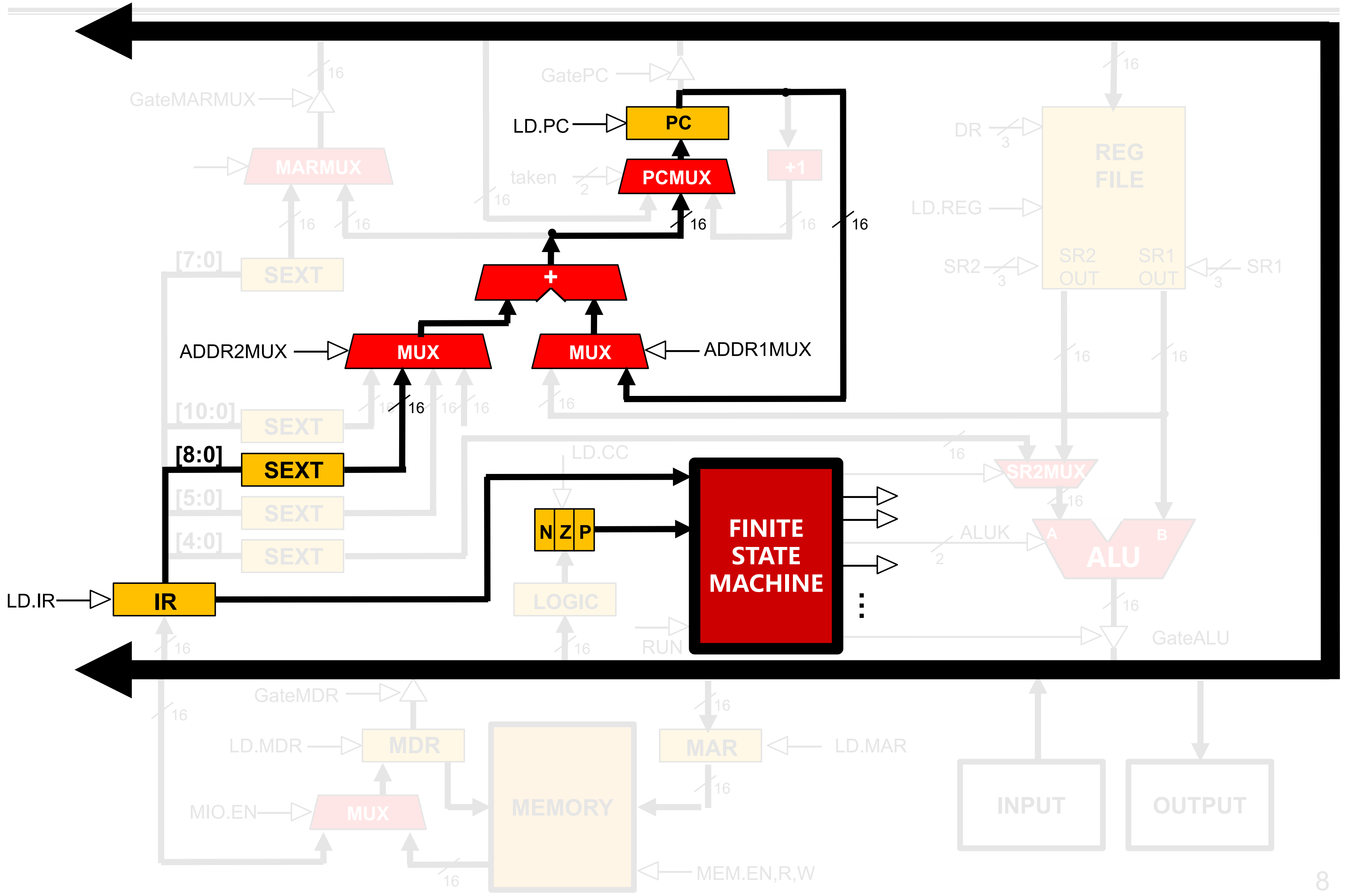
BR (PC-Relative)



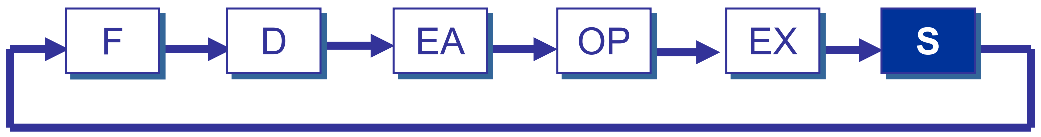


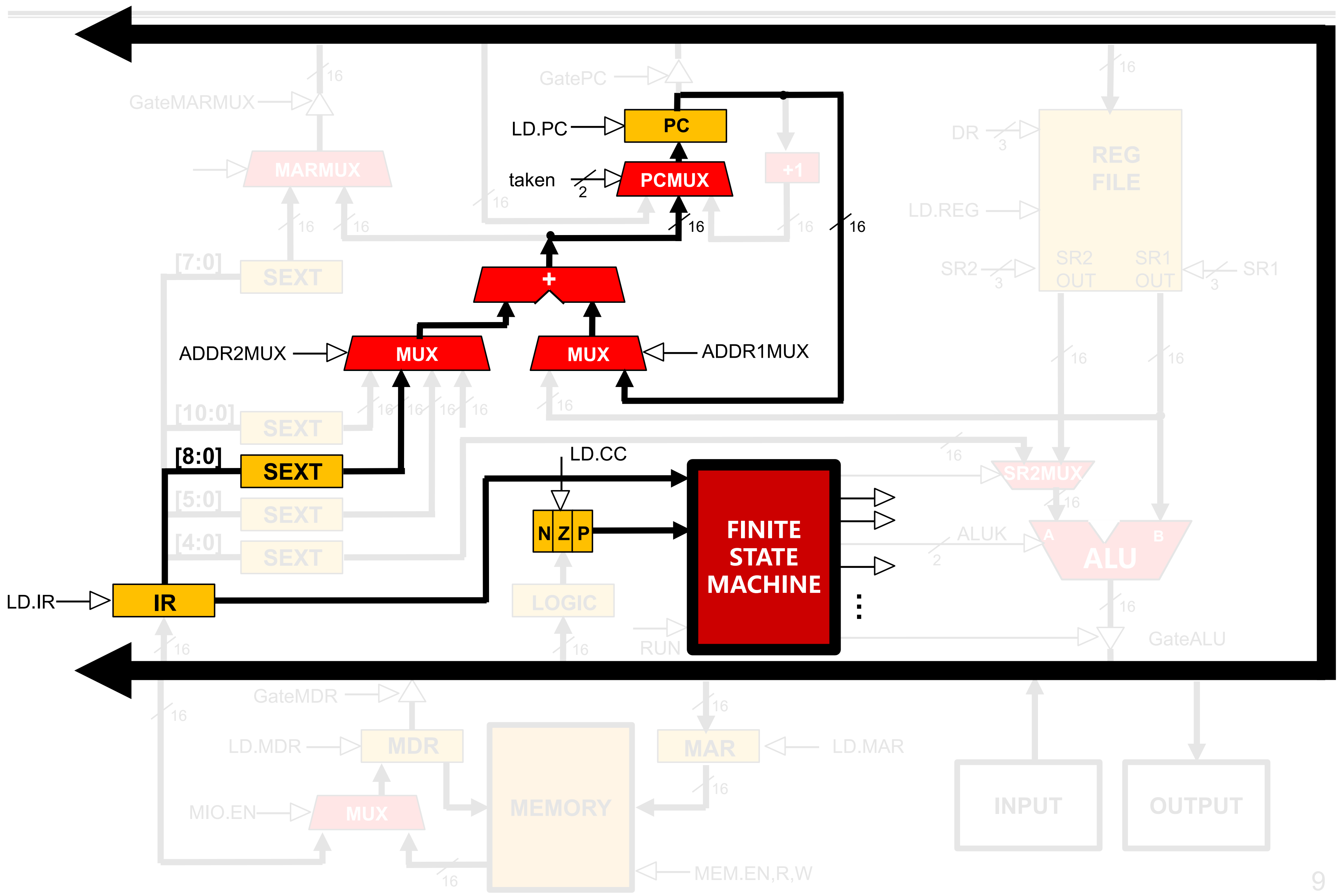
BR (PC-Relative)





BR (PC-Relative)





BR (PC-Relative)

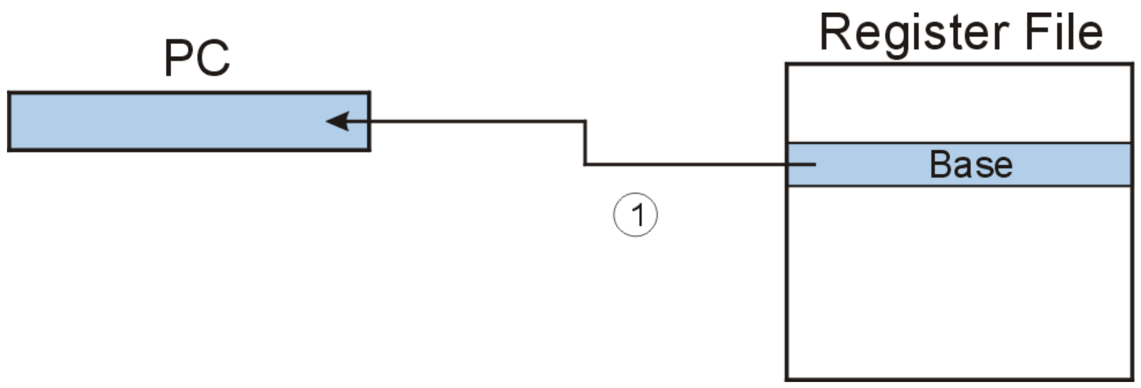
- **Check**
  - BR_{nzp} x4101 ; if (n=1 or z=1 or p=1) , JMP x4101
  - BR_{n} x4101 ; if (n=1)
  - BR_{z} x4101 ; if (z=1)
  - BR_{p} x4101 ; if (p=1)
  - BR_{nz} x4101 ; if (n=1 or z=1)
  - BR_{np} x4101 ; if (n=1 or p=1)
  - BR_{zp} x4101 ; if (z=1 or p=1)
  - BR x4101 ; PC=PC+1
- **Set**
  - If DR < 0, set N=1 and Z=0 and P=0
  - If DR = 0, set N=0 and Z=1 and P=0
  - If DR > 0, set N=0 and Z=0 and P=1

JMP (Register)

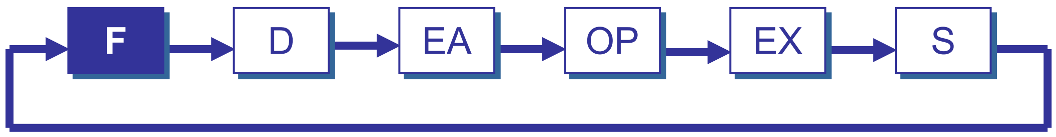
Jump is an unconditional branch -- *always* taken.

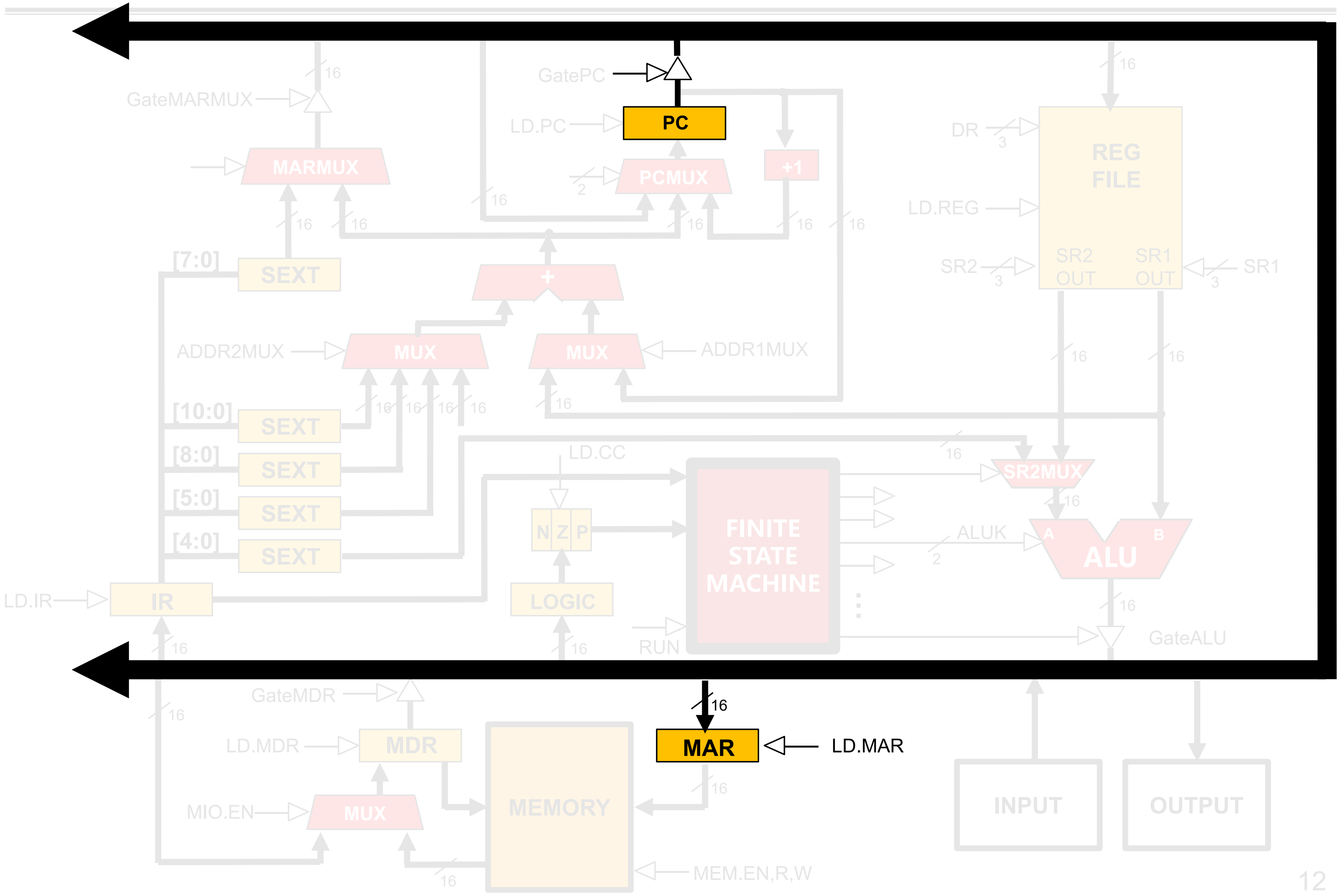
- **Target address is the contents of a register.**
- **Allows any target address.**

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

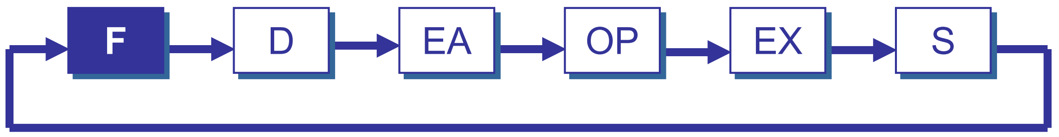


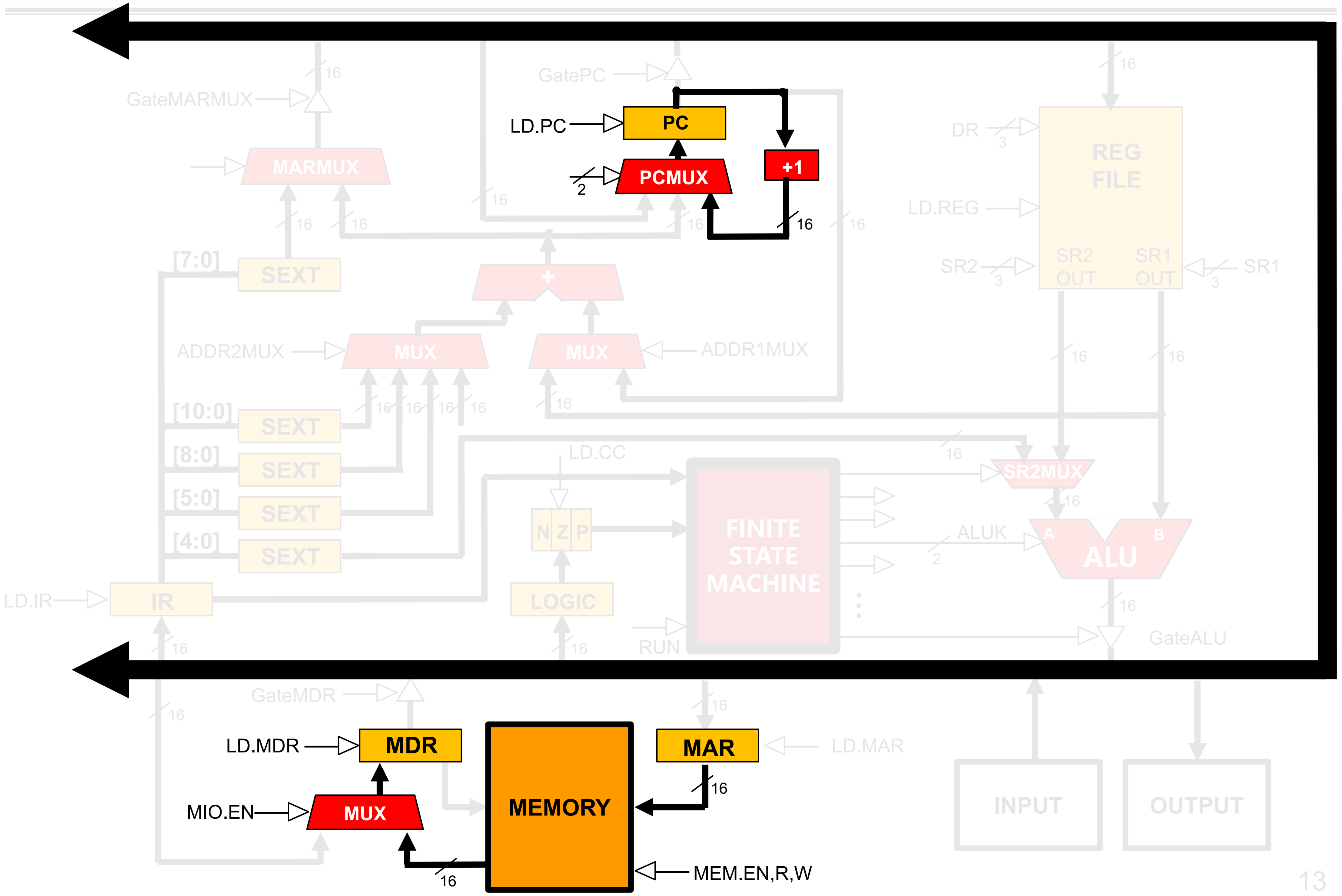
JMP (Register)



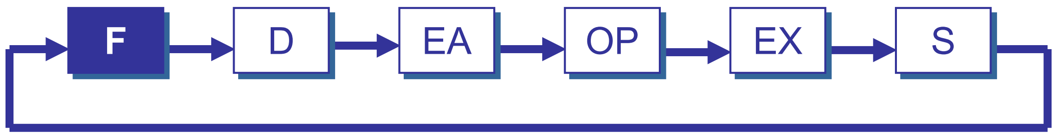


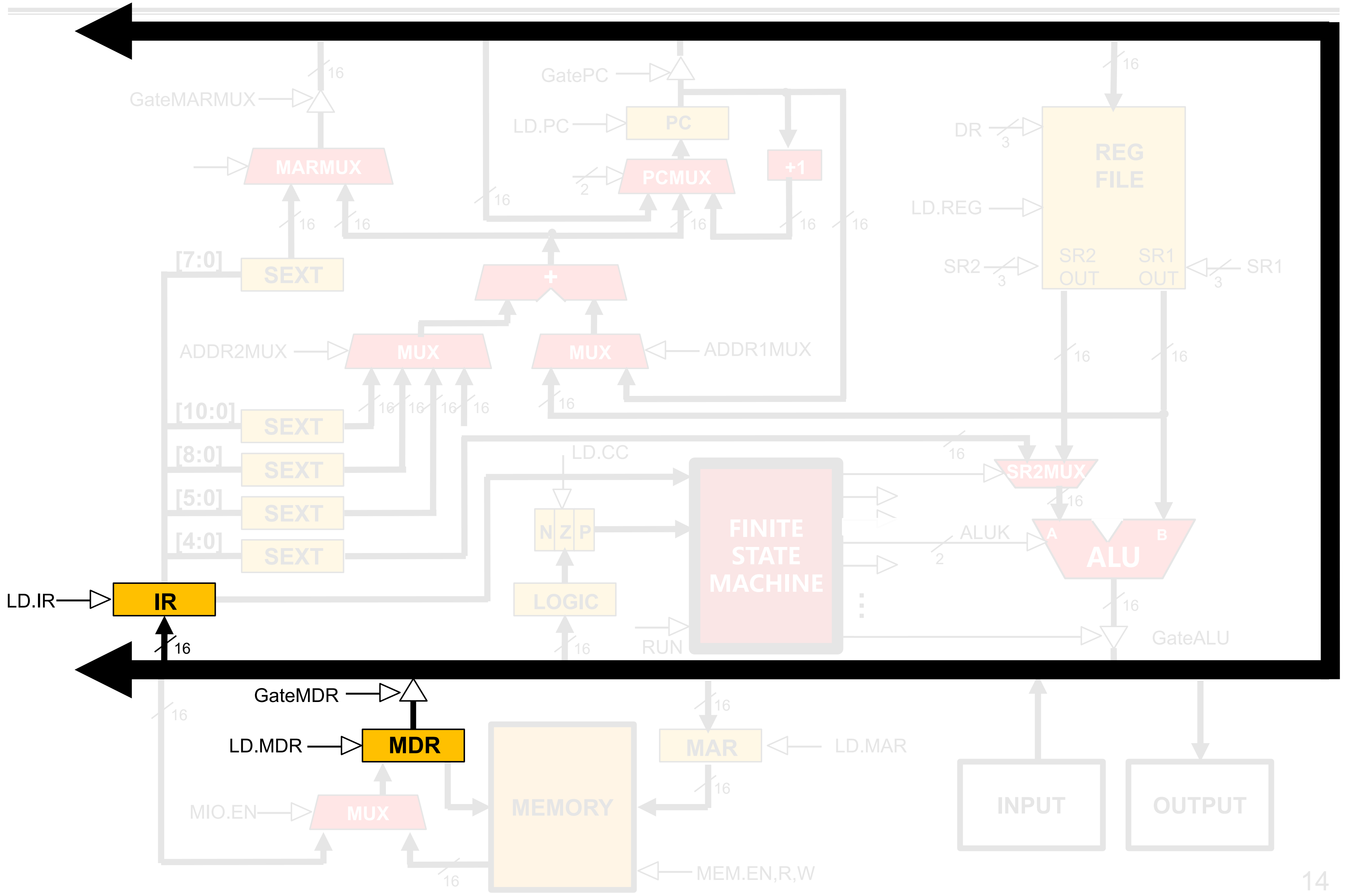
JMP (Register)



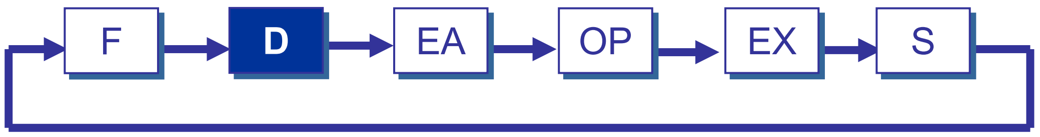


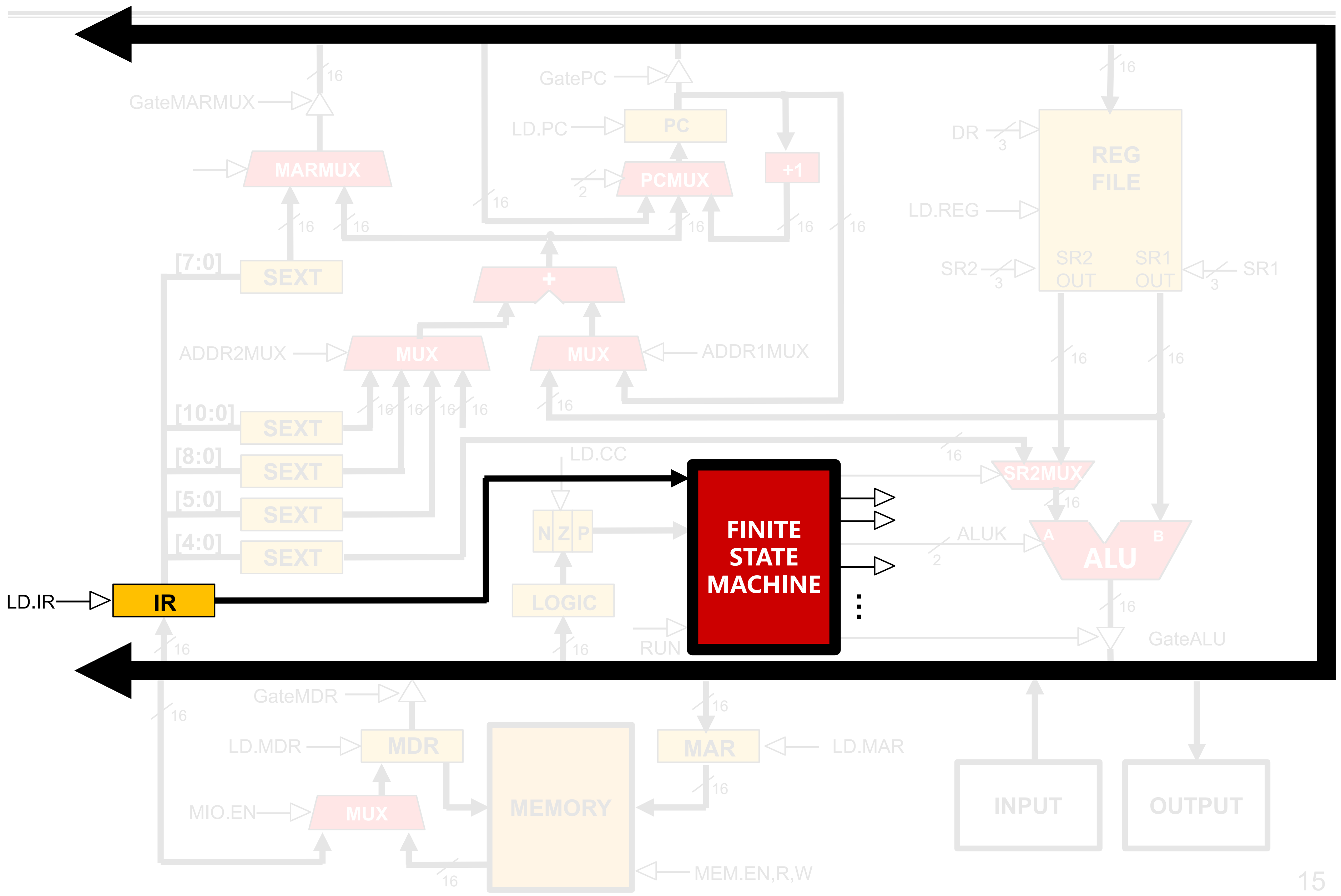
JMP (Register)



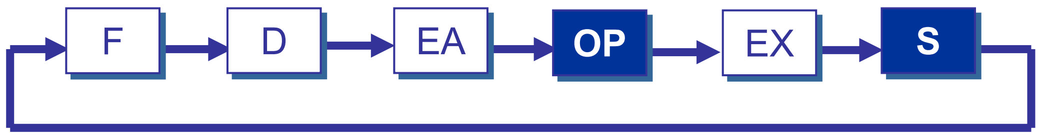


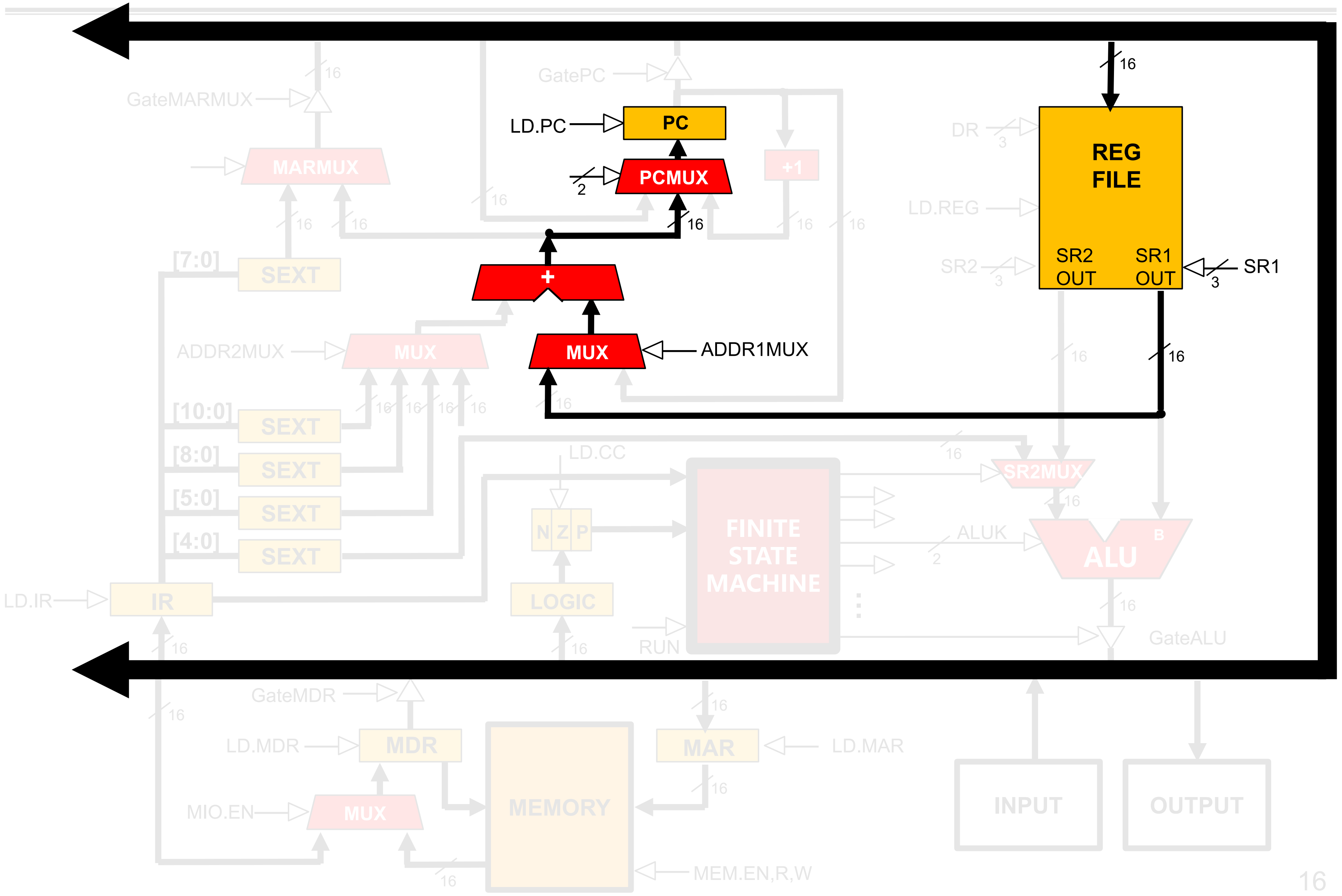
JMP R7(Register)





JMP R7(Register)





TRAP

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRAP	1	1	1	1	0	0	0	0	trapvect8							

Calls a service routine, identified by 8-bit "trap vector."

vector	*routine*
`x23`	input a character from the keyboard
`x21`	output a character to the monitor
`x25`	halt the program

Example:

TRAP x23

; Directs the operating system to execute the IN system call.

; The starting address of this system call is contained in memory location x0023.

TRAP

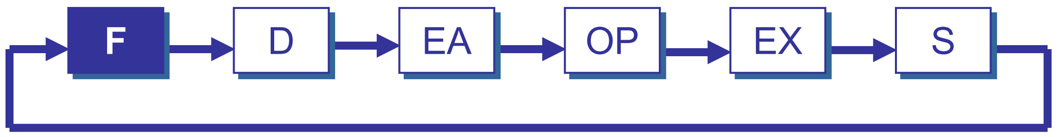
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRAP	1	1	1	1	0	0	0	0	trapvect8							

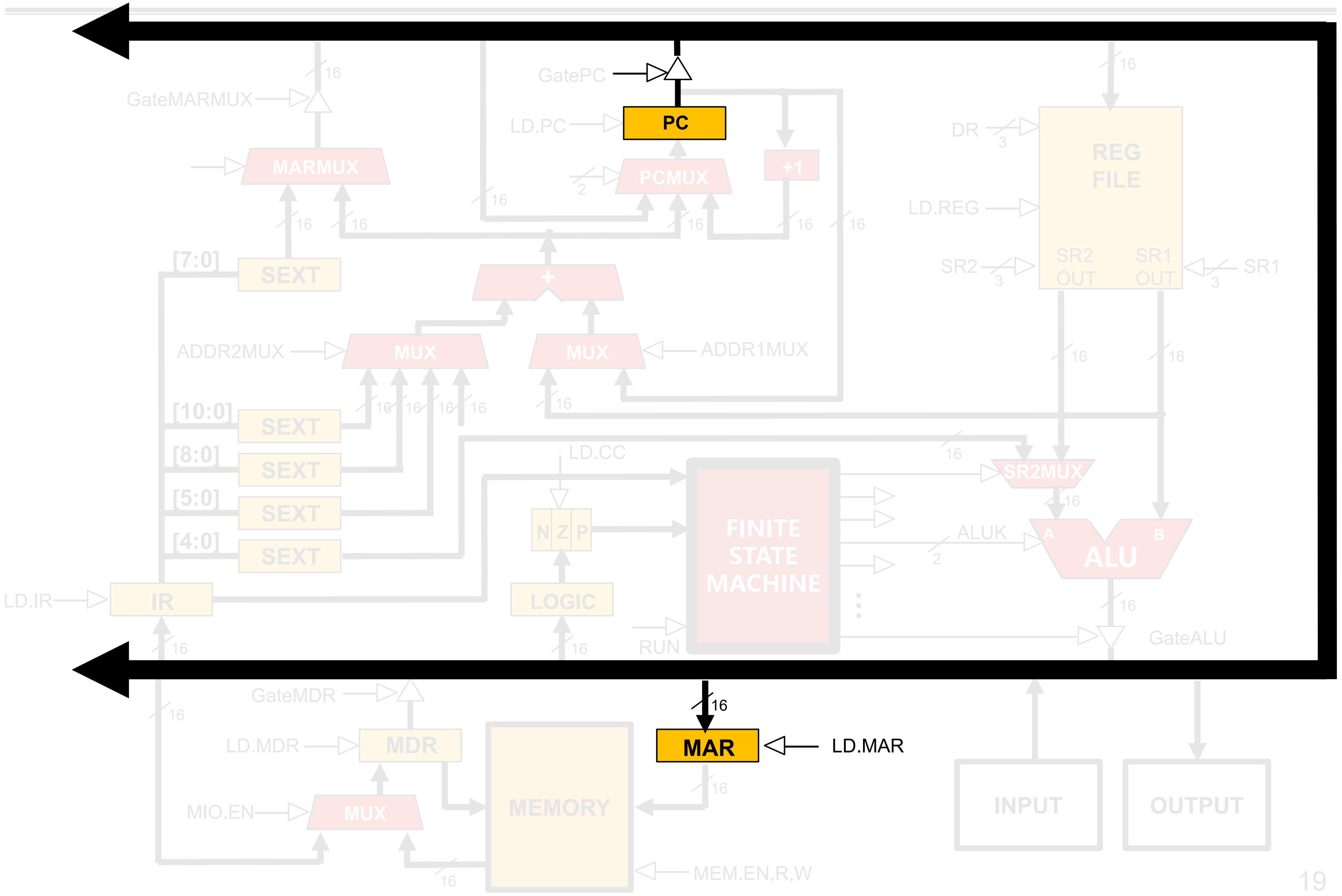
Calls a service routine, identified by 8-bit "trap vector."

vector	*routine*
x23	input a character from the keyboard
x21	output a character to the monitor
x25	halt the program

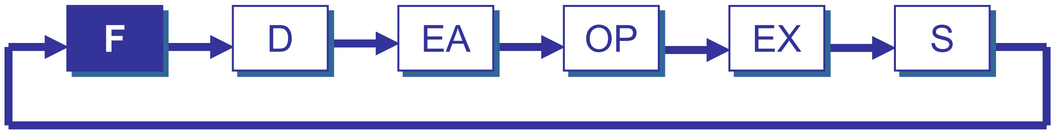
When routine is done,
PC is set to the instruction following TRAP.
(We' ll talk about how this works later.)

TRAP

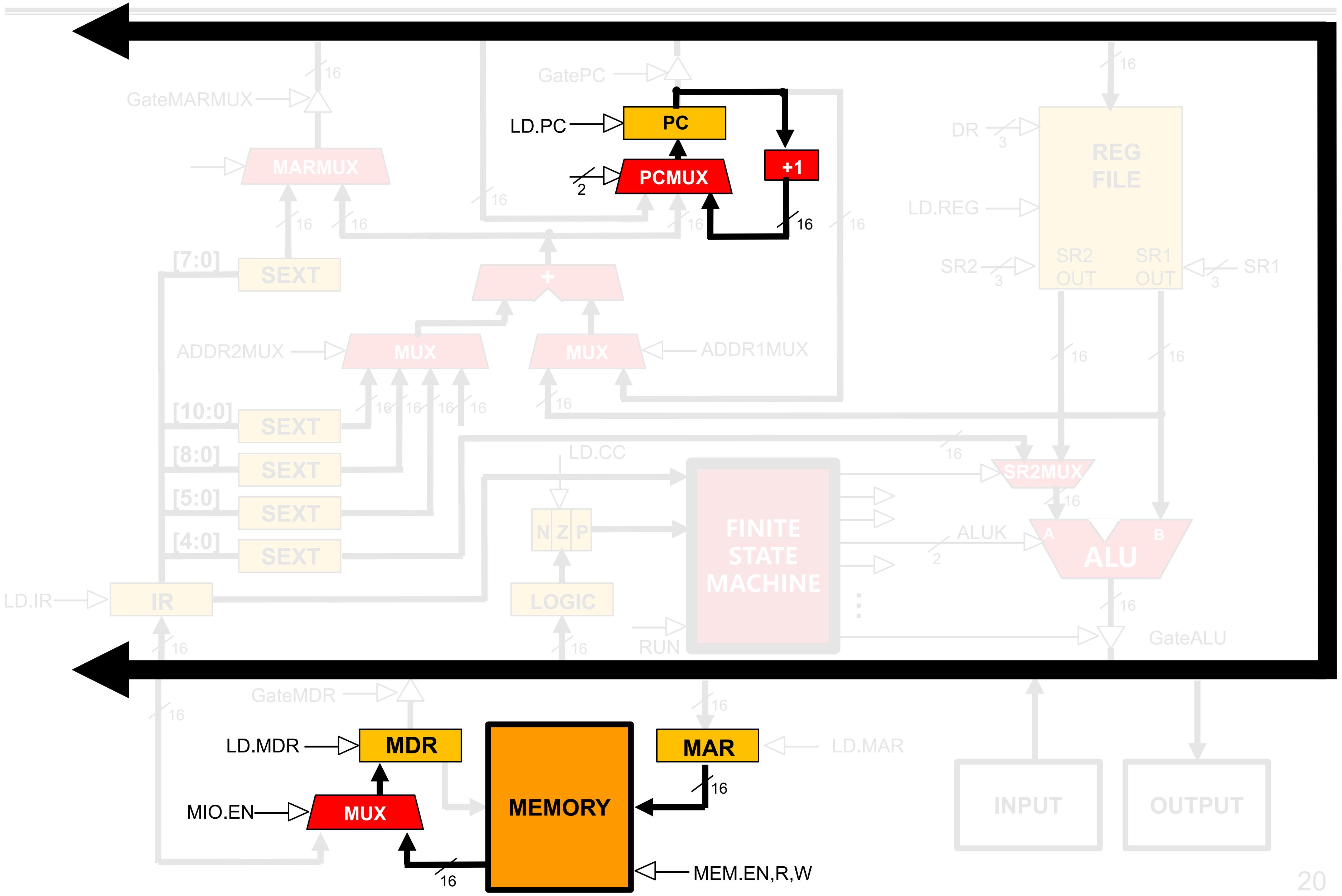

F
D
EA
OP
EX
S


GatePC
LD.PC
PC
PCMUX
+1
GateMARMUX
MARMUX
16
2
[7:0]
SEXT
+
ADDR2MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.IR
IR
LD.CC
N Z P
LOGIC
FINITE STATE MACHINE
RUN
DR
LD.REG
REG FILE
SR2
SR1
SR2 OUT
SR1 OUT
SR2MUX
ALUK
ALU
A
B
GateALU
3
GateMDR
LD.MDR
MDR
MIO.EN
MEMORY
MEM.EN,R,W
MAR
LD.MAR
INPUT
OUTPUT

TRAP

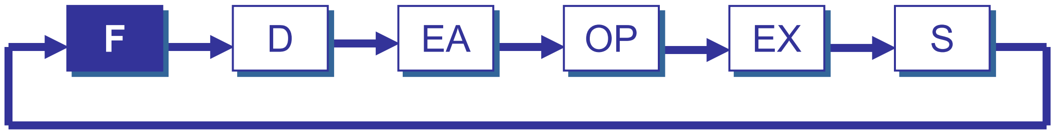


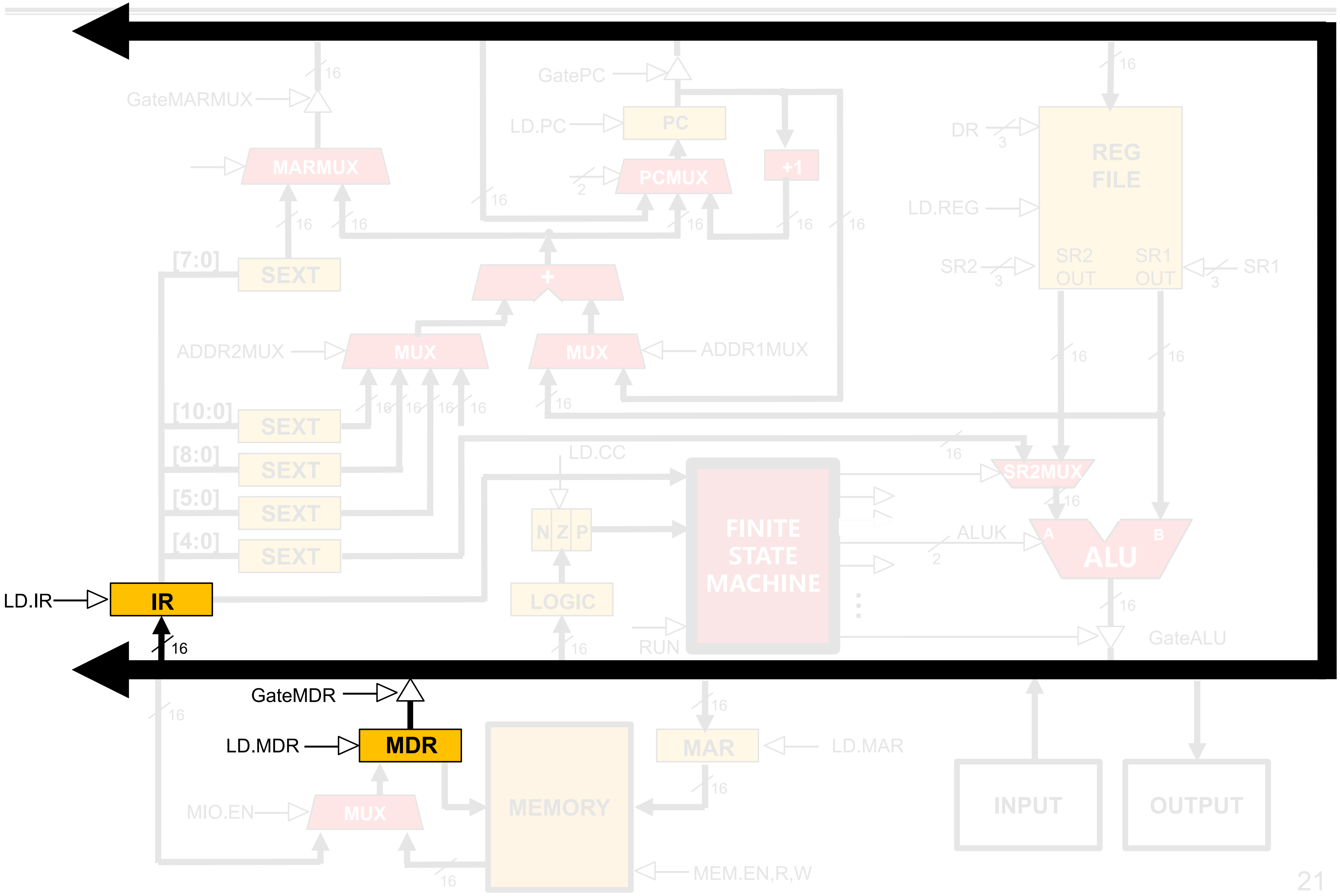
F
D
EA
OP
EX
S



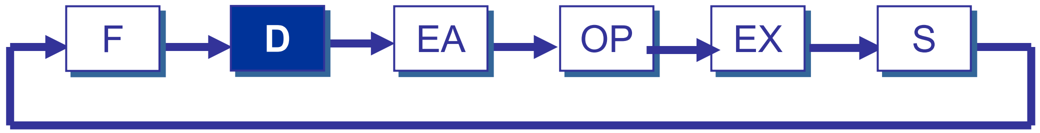
LD.PC
PC
PCMUX
2
+1
16
GatePC
GateMARMUX
MARMUX
SEXT
[7:0]
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.IR
IR
LD.CC
N Z P
LOGIC
FINITE STATE MACHINE
RUN
REG FILE
DR
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
SR2MUX
ALUK
ALU
GateALU
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MAR
LD.MAR
MEM.EN,R,W
INPUT
OUTPUT

TRAP

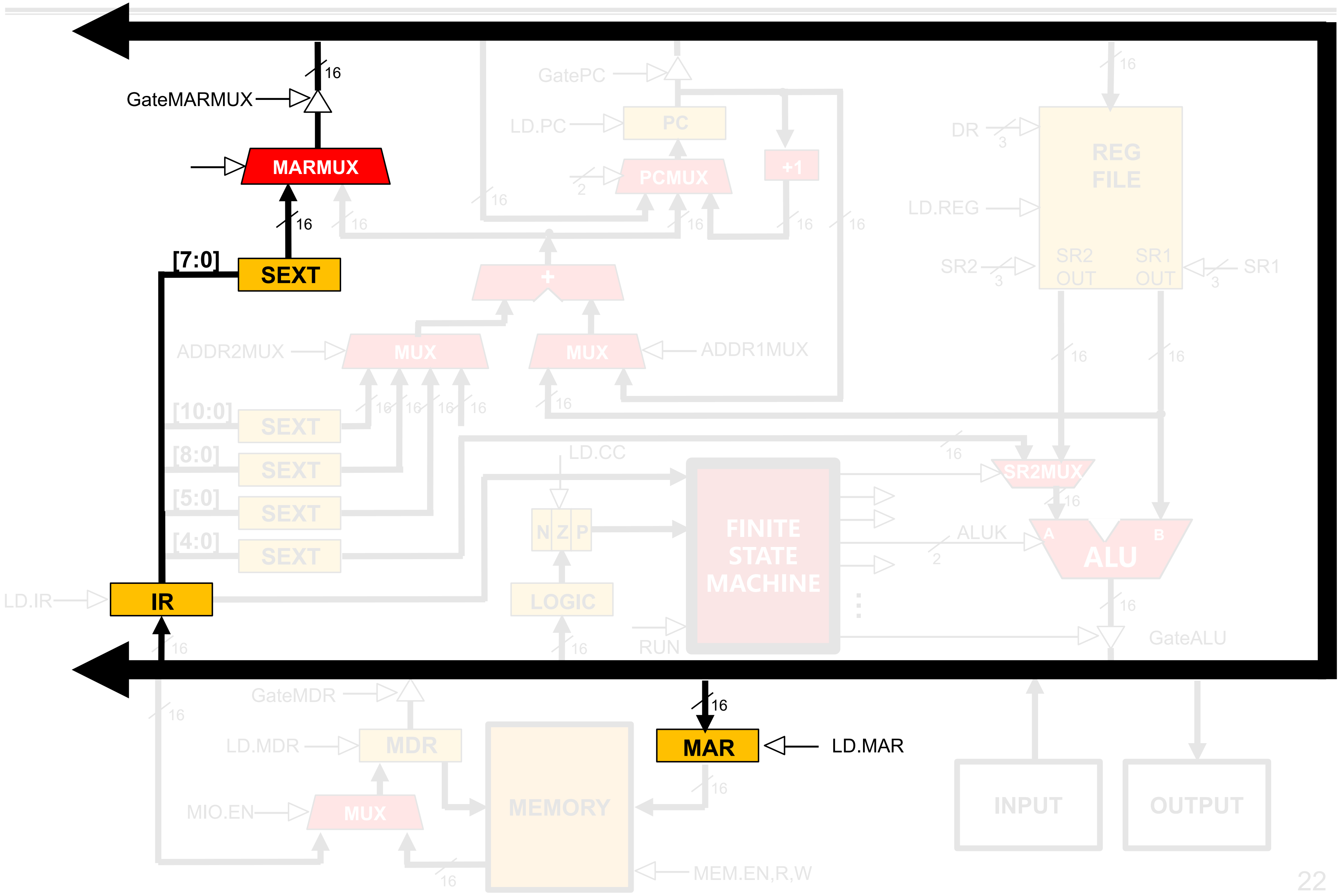

F
D
EA
OP
EX
S


LD.IR
IR
16
GateMDR
LD.MDR
MDR
GateMARMUX
MARMUX
GatePC
LD.PC
PC
PCMUX
+1
REG FILE
DR
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
SEXT
[7:0]
[10:0]
[8:0]
[5:0]
[4:0]
ADDR2MUX
MUX
ADDR1MUX
LD.CC
N Z P
FINITE STATE MACHINE
SR2MUX
ALUK
ALU
LOGIC
RUN
GateALU
MIO.EN
MEMORY
MAR
LD.MAR
MEM.EN,R,W
INPUT
OUTPUT

TRAP

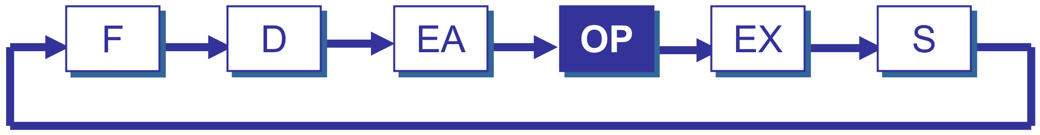


F
D
EA
OP
EX
S

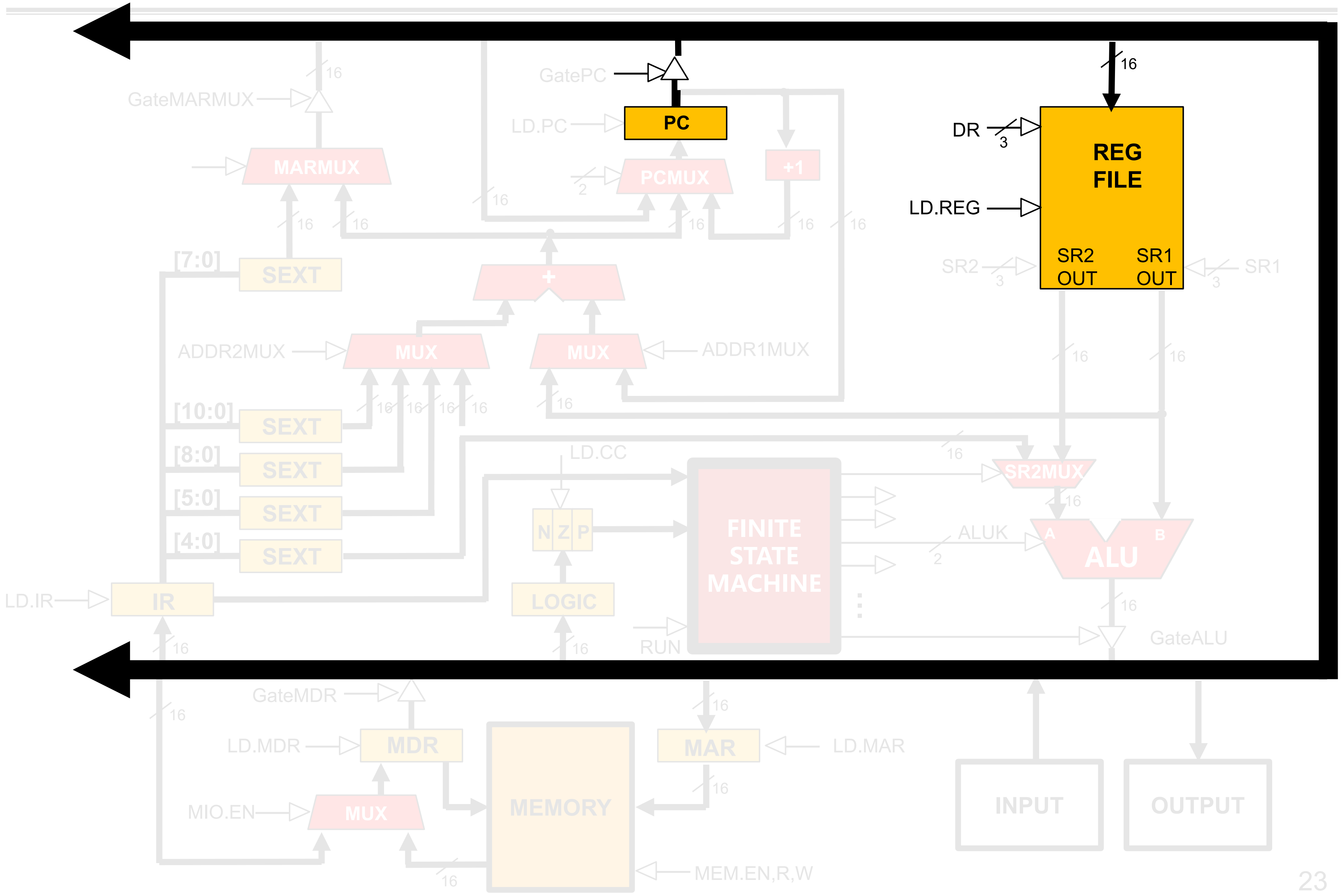


GateMARMUX
16
MARMUX
16
[7:0]
SEXT
IR
LD.IR
16
MAR
LD.MAR
16
GatePC
LD.PC
PC
PCMUX
+1
REG FILE
DR
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
ADDR2MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
SEXT
LD.CC
N
Z
P
LOGIC
FINITE STATE MACHINE
SR2MUX
ALUK
ALU
A
B
GateALU
RUN
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MEM.EN,R,W
INPUT
OUTPUT

TRAP

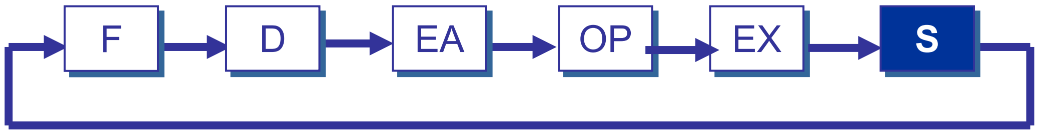


F
D
EA
OP
EX
S

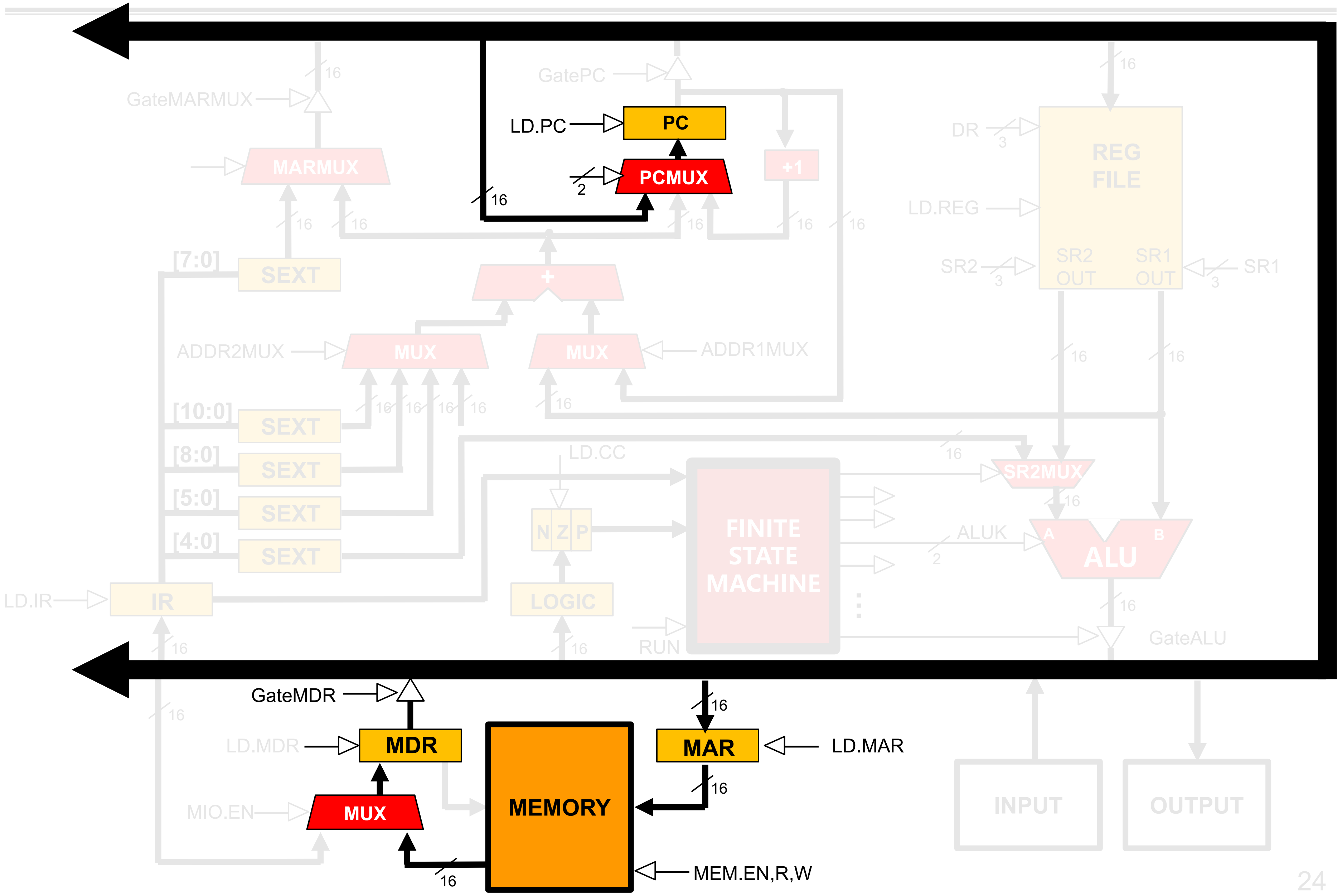


GateMARMUX
16
MARMUX
GatePC
LD.PC
PC
PCMUX
2
+1
16
DR
3
LD.REG
REG FILE
SR2 OUT
SR1 OUT
SR2
SR1
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.CC
N Z P
FINITE STATE MACHINE
SR2MUX
ALUK
ALU
A
B
GateALU
LD.IR
IR
LOGIC
RUN
GateMDR
LD.MDR
MDR
MEMORY
MAR
LD.MAR
MIO.EN
MUX
MEM.EN,R,W
INPUT
OUTPUT

TRAP

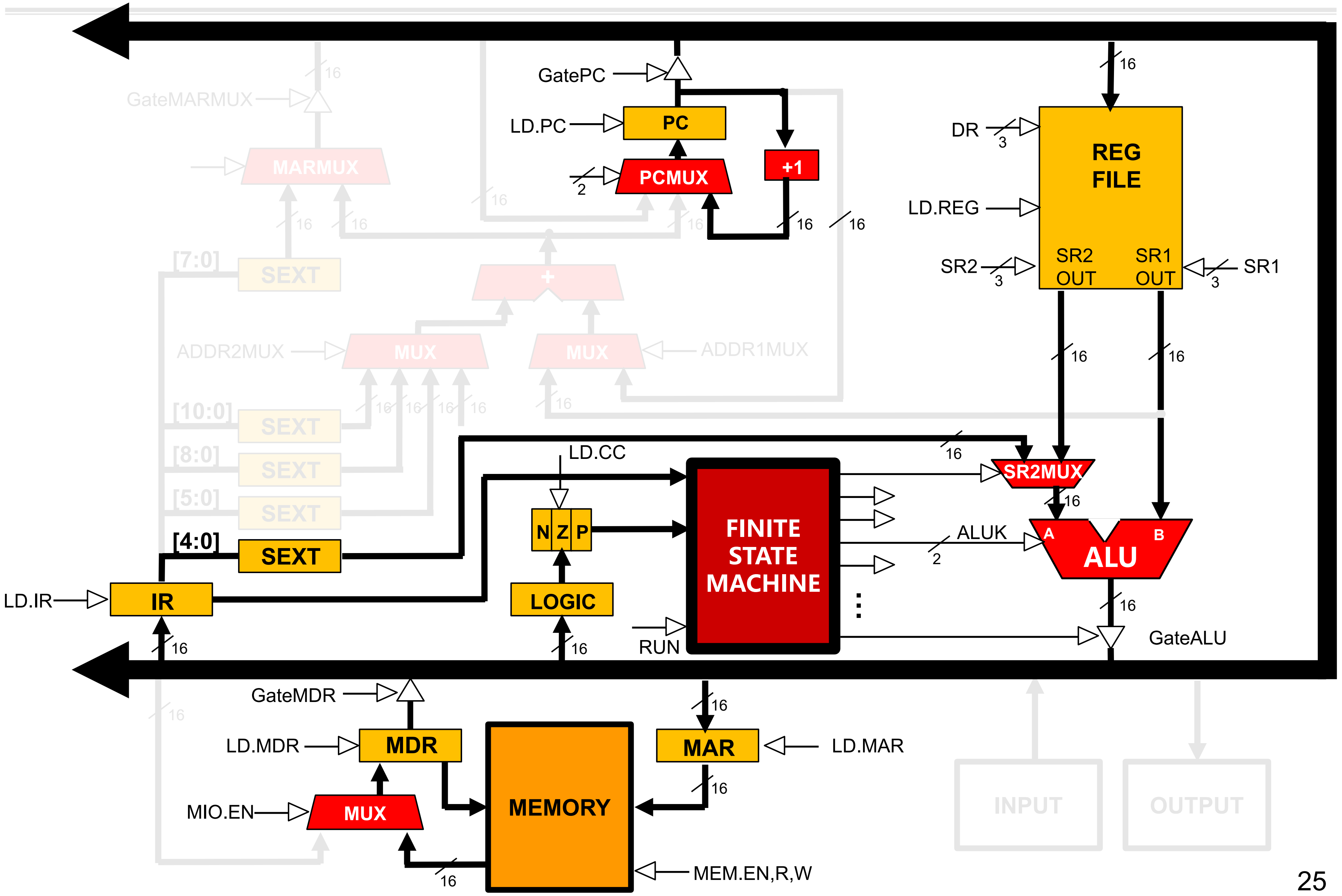


F
D
EA
OP
EX
S

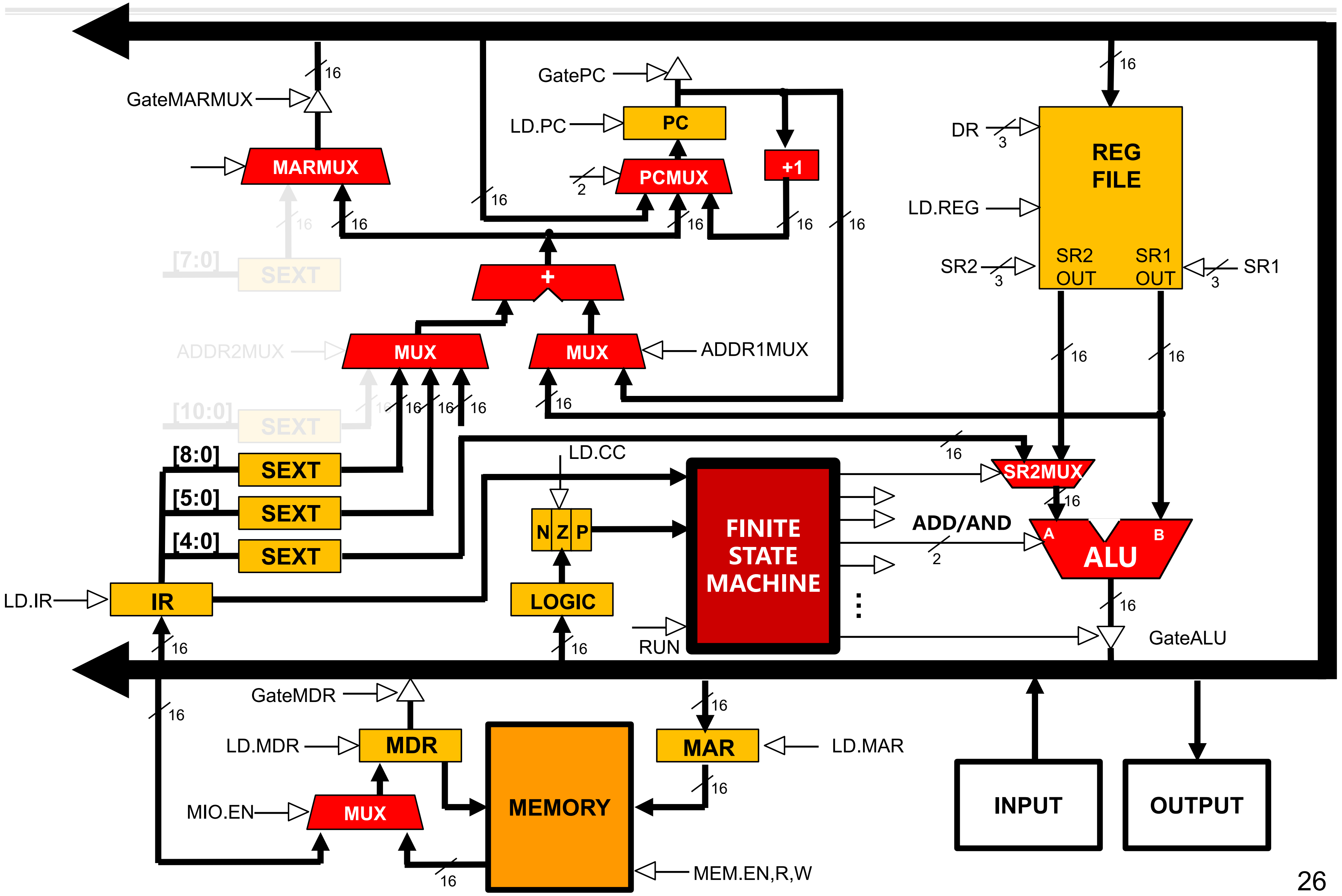


LD.PC
PC
PCMUX
2
16
GatePC
GateMARMUX
MARMUX
+1
SEXT
[7:0]
[10:0]
[8:0]
[5:0]
[4:0]
ADDR2MUX
ADDR1MUX
MUX
REG FILE
DR
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
SR2MUX
ALU
ALUK
GateALU
LD.CC
N Z P
LOGIC
FINITE STATE MACHINE
RUN
LD.IR
IR
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MAR
LD.MAR
MEM.EN,R,W
INPUT
OUTPUT

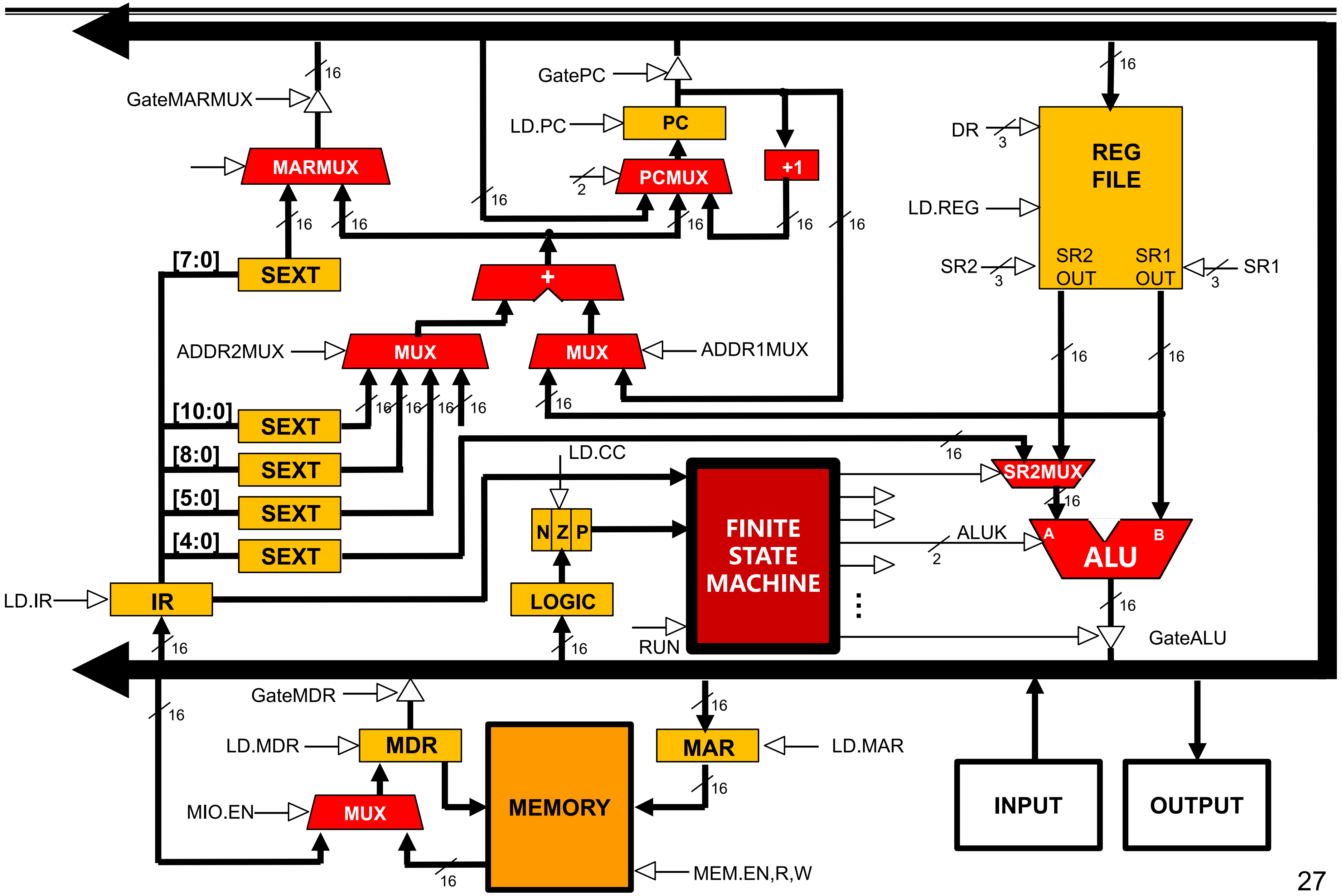
LC-3 Data Path After Operate Instruction



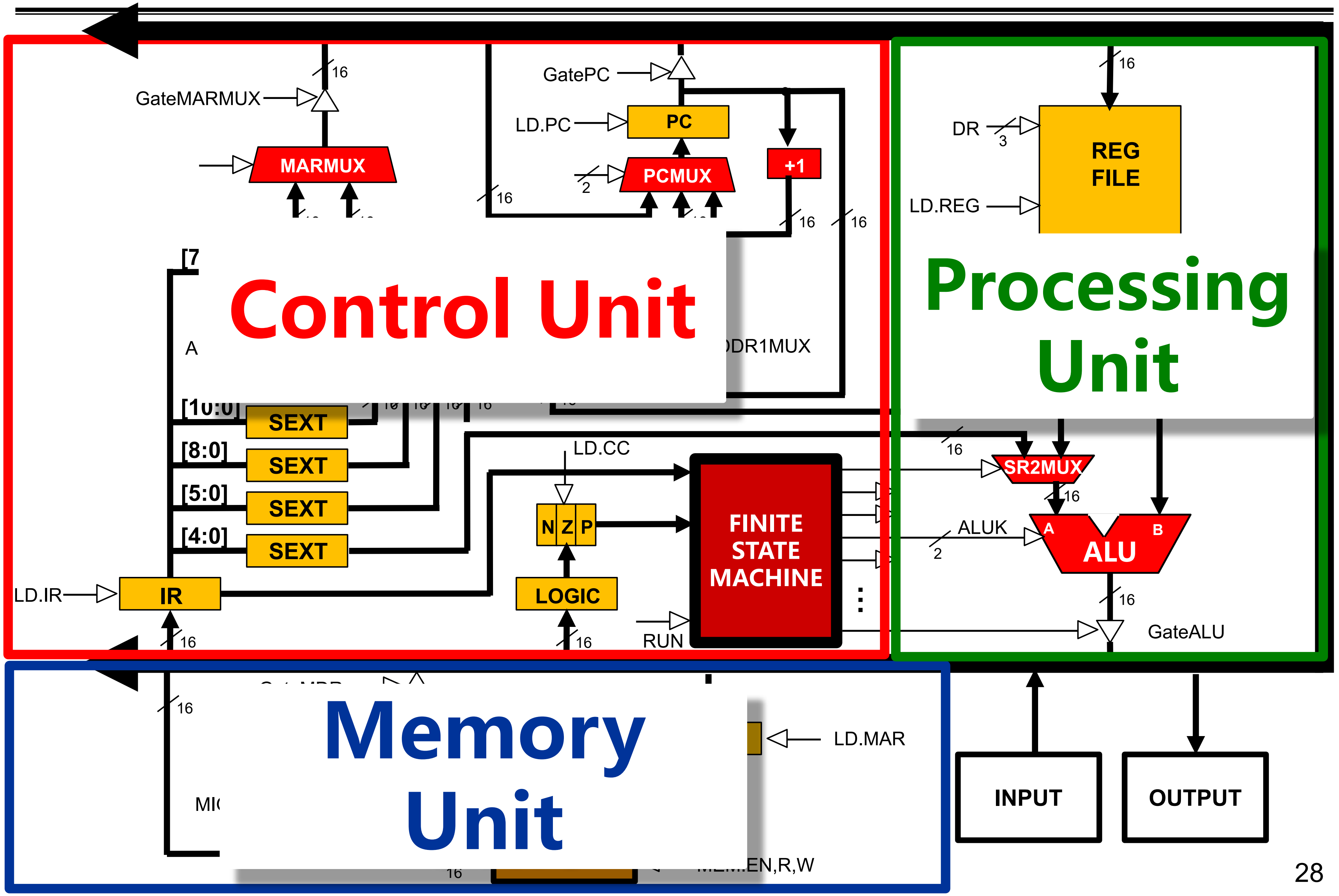
LC-3 Data Path After Load/Store Instruction



LC-3 Data Path After Control Instruction



LC-3 Data Path


GateMARMUX
16
MARMUX
GatePC
LD.PC
PC
PCMUX
2
+1
16
16
16
Control Unit
A
[7
DR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
SEXT
SEXT
SEXT
SEXT
LD.CC
N Z P
LOGIC
FINITE STATE MACHINE
RUN
LD.IR
IR
16
16
Processing Unit
DR
3
LD.REG
REG FILE
16
SR2MUX
16
ALUK
2
A
B
ALU
16
GateALU
Memory Unit
16
LD.MAR
MI
MEM.EN,R,W
16
INPUT
OUTPUT