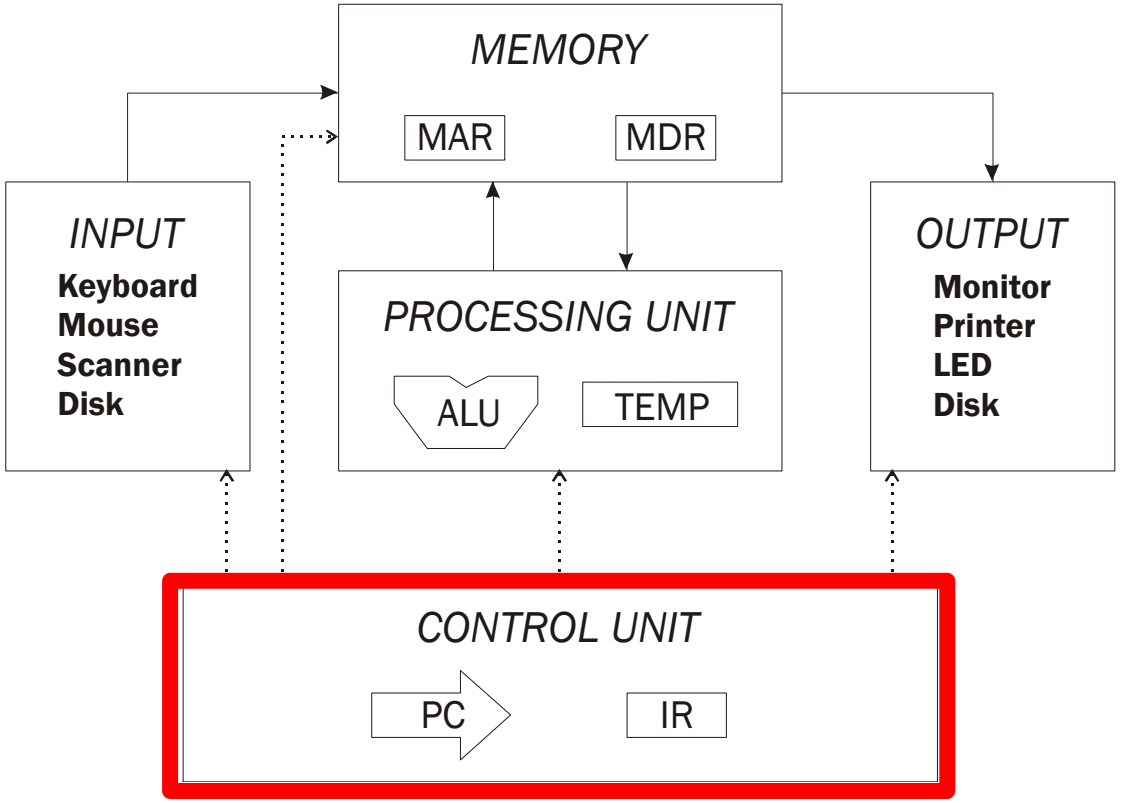
310-2202 โครงสร้างของระบบคอมพิวเตอร์ (Computer Organization)

Topic 5: The LC-3 Instructions: Control Instructions

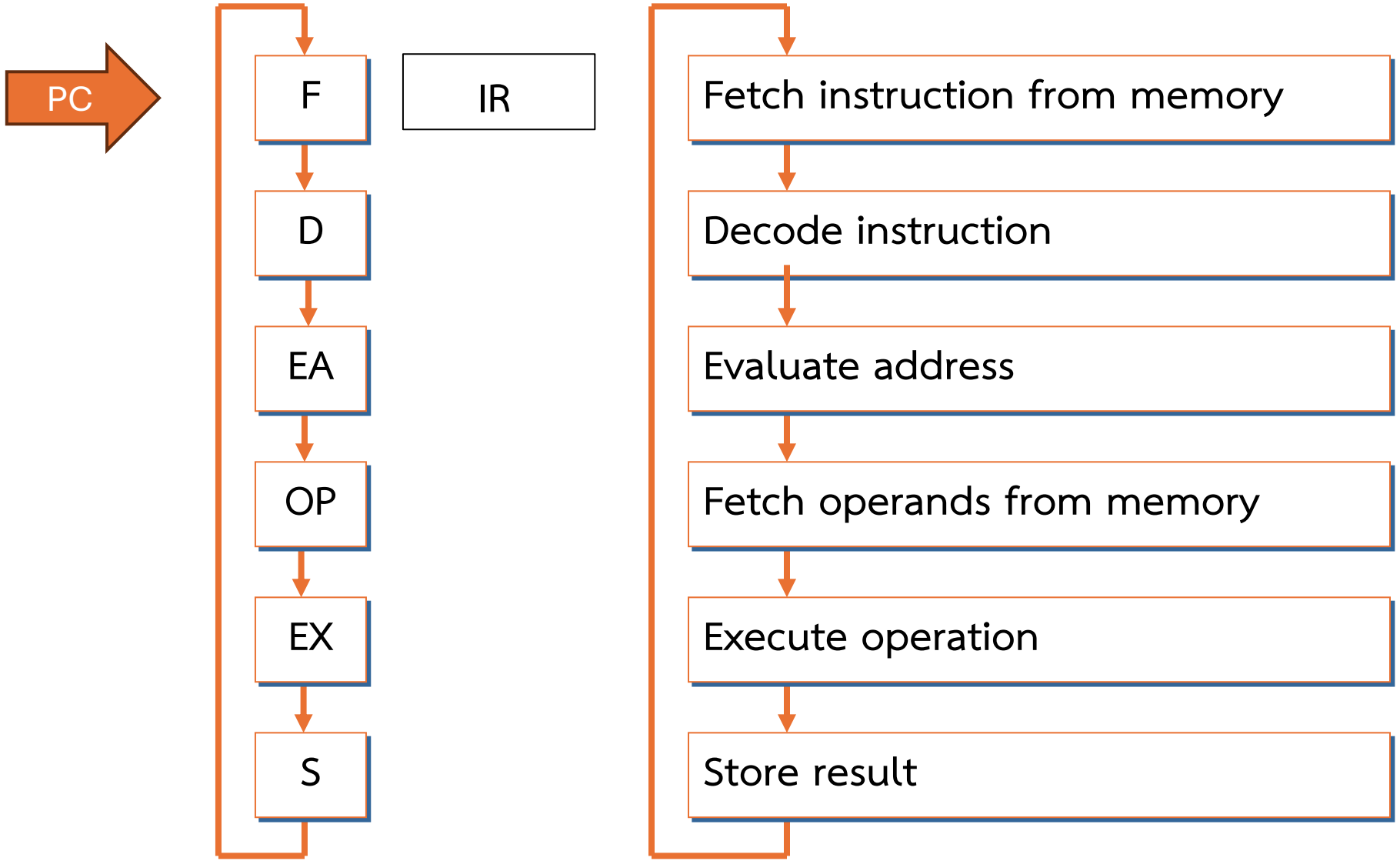
Damrongrit Setsirichok

Topic

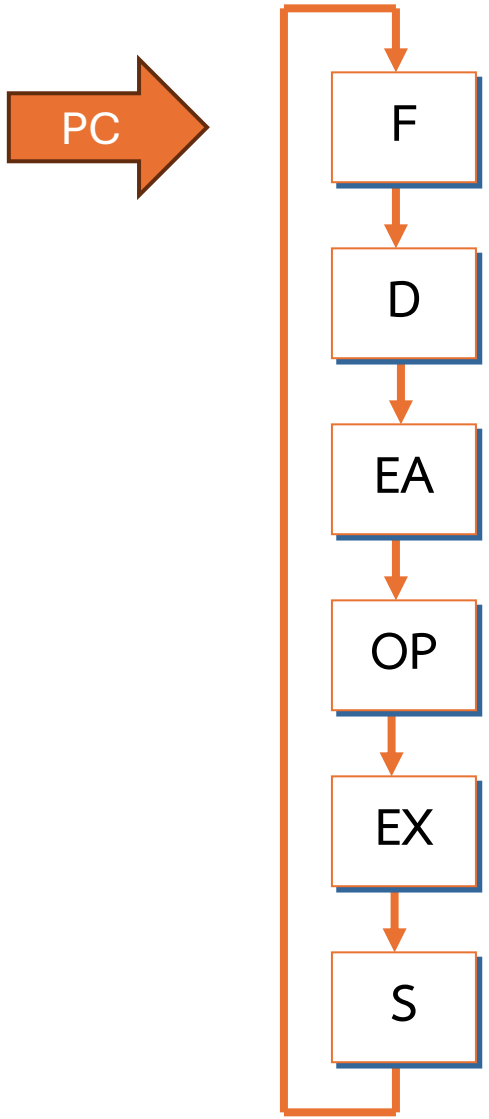
- LC-3 Control Instructions
- Conditional Branch Instruction and Loop Control Example
- Jump & TRAP Instruction



Instruction Processing: State Transition



Instruction Processing: Finite State Automata



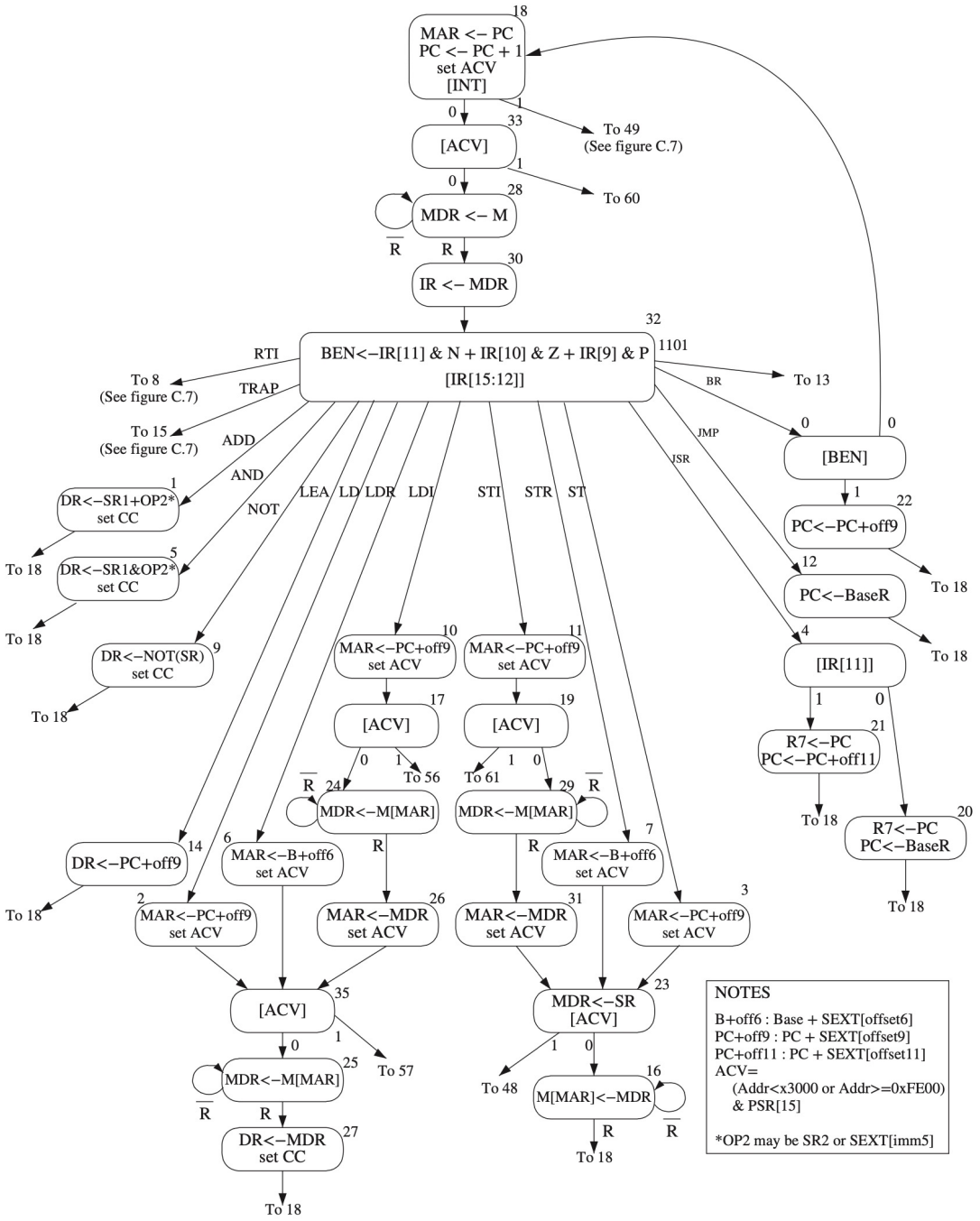
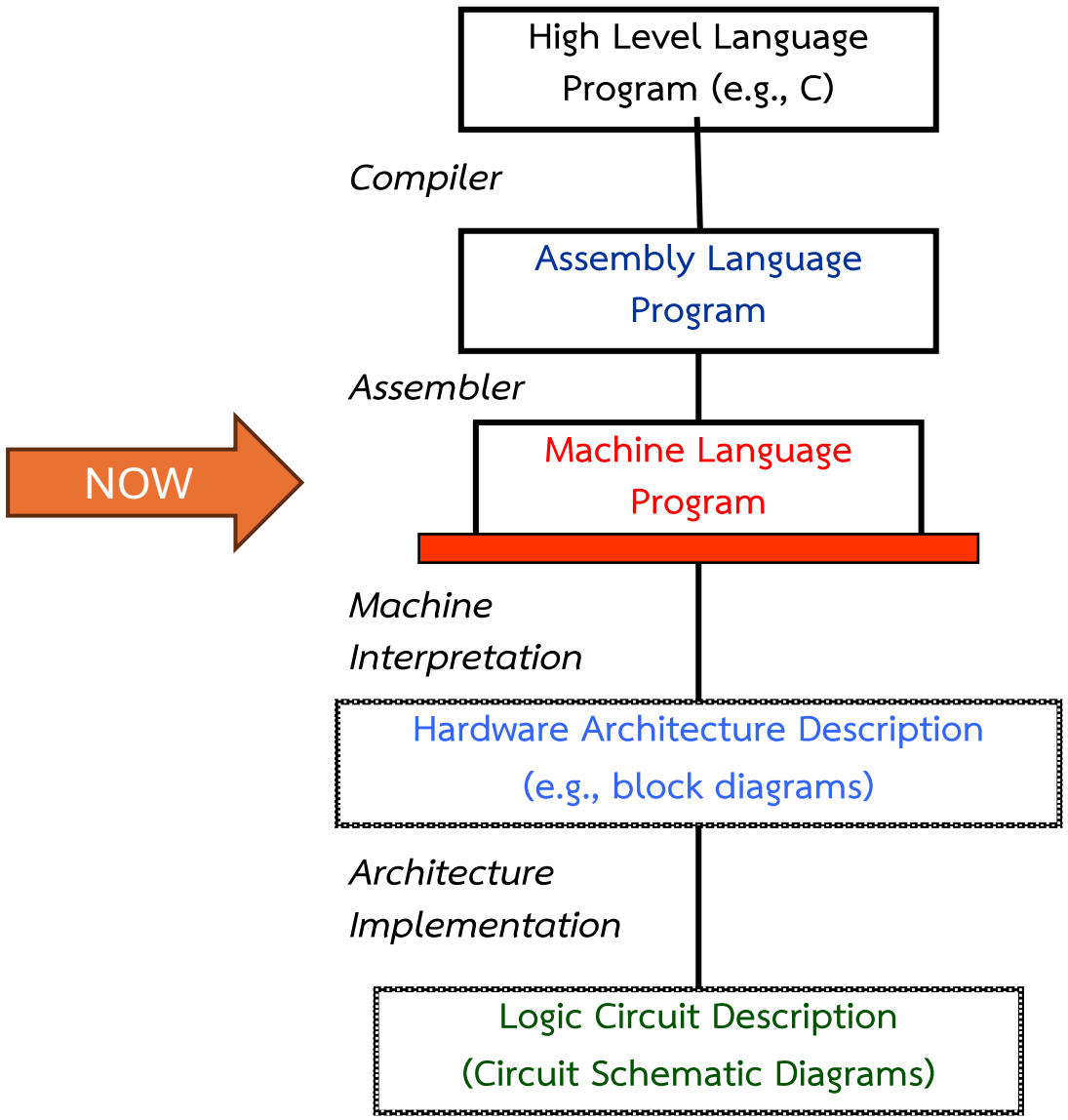


Figure C.2 A state machine for the LC-3.

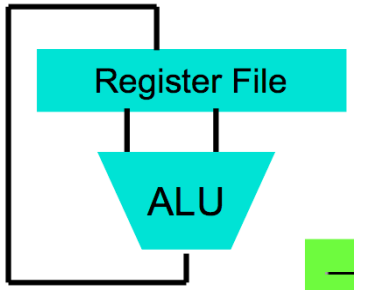
How do we get the electrons to do the work?

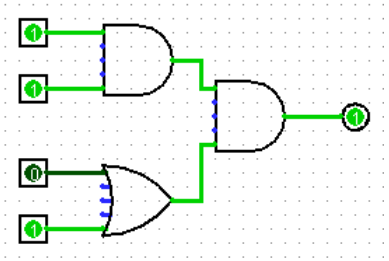



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
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 Control Instructions

Control Instructions

Conditional Branch

- branch is ***taken*** if a specified condition is true
  - signed offset is added to PC to yield new PC
- else, the branch is *not taken*
  - PC is not changed, points to the next sequential instruction

Unconditional Branch (or Jump)

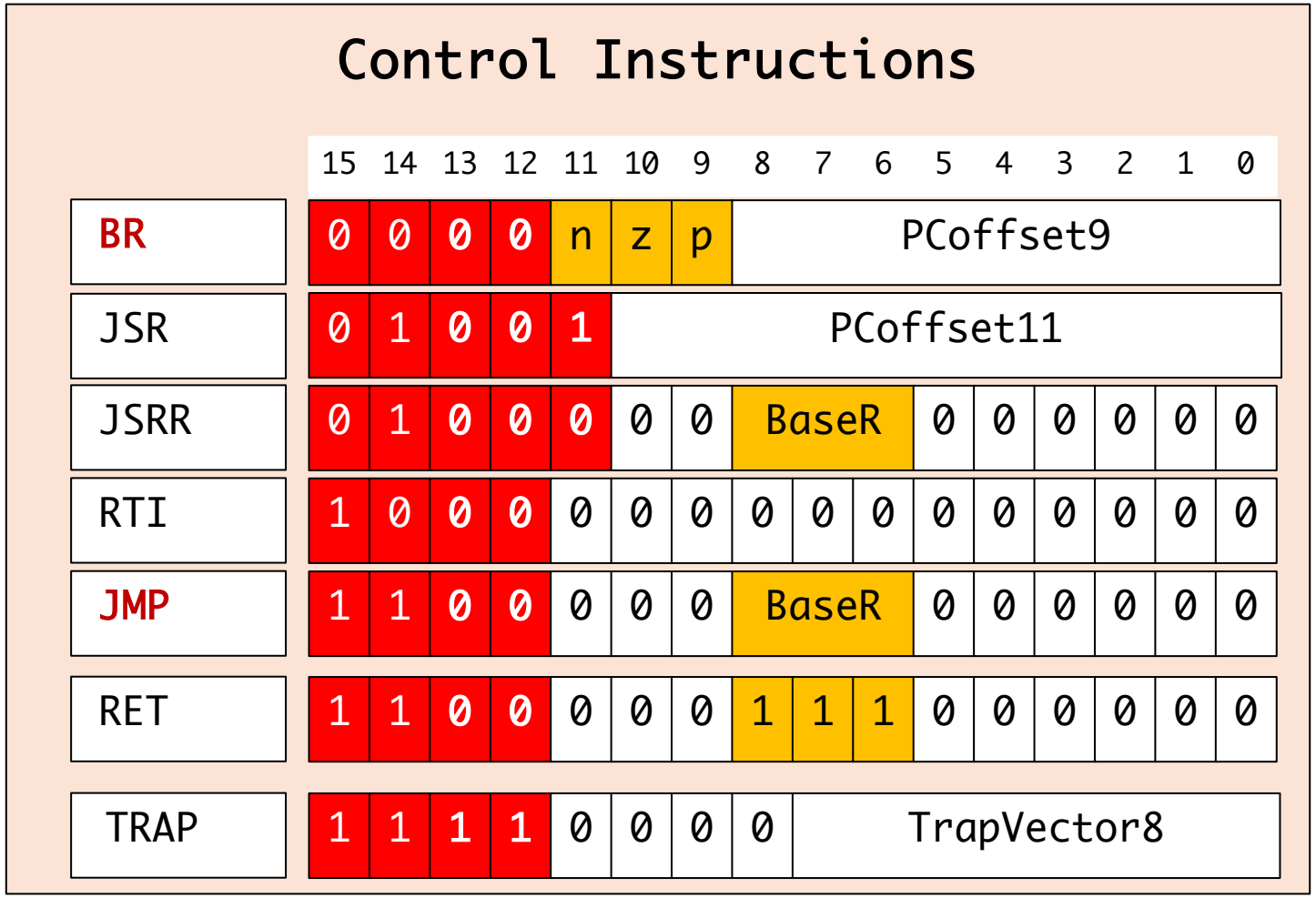
- **always changes the PC**

TRAP

- changes PC to the address of an OS “service routine”
- routine will return control to the next instruction (after TRAP)

LC-3 ISA

Control Instructions



Condition Codes

- LC-3 has three condition code registers:
  - N -- negative
  - Z -- zero
  - P -- positive (greater than zero)

- Set by any instruction that writes a value to a register (ADD, AND, NOT, LD, LDR, LDI, LEA)

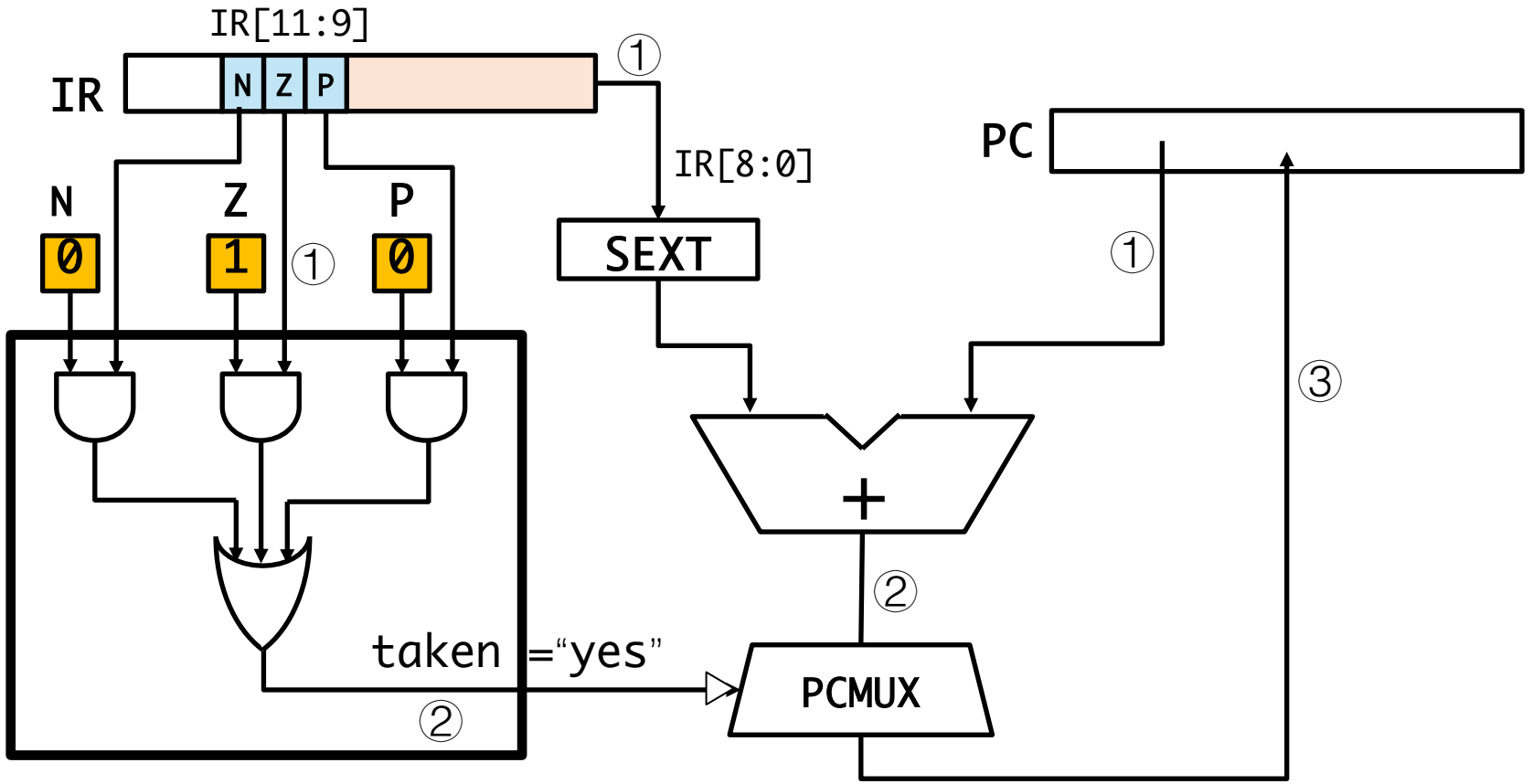
- Exactly one will be set at all times
  - Based on the last instruction that altered a register

LC-3 Conditional Branch Instruction and Loop Control Example

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[7:0]) to current PC.
  - *** PC has already **been incremented** by FETCH stage.
  - *** Target must be within 256 words of BR instruction.
- If the branch is not taken, the next sequential instruction is executed.

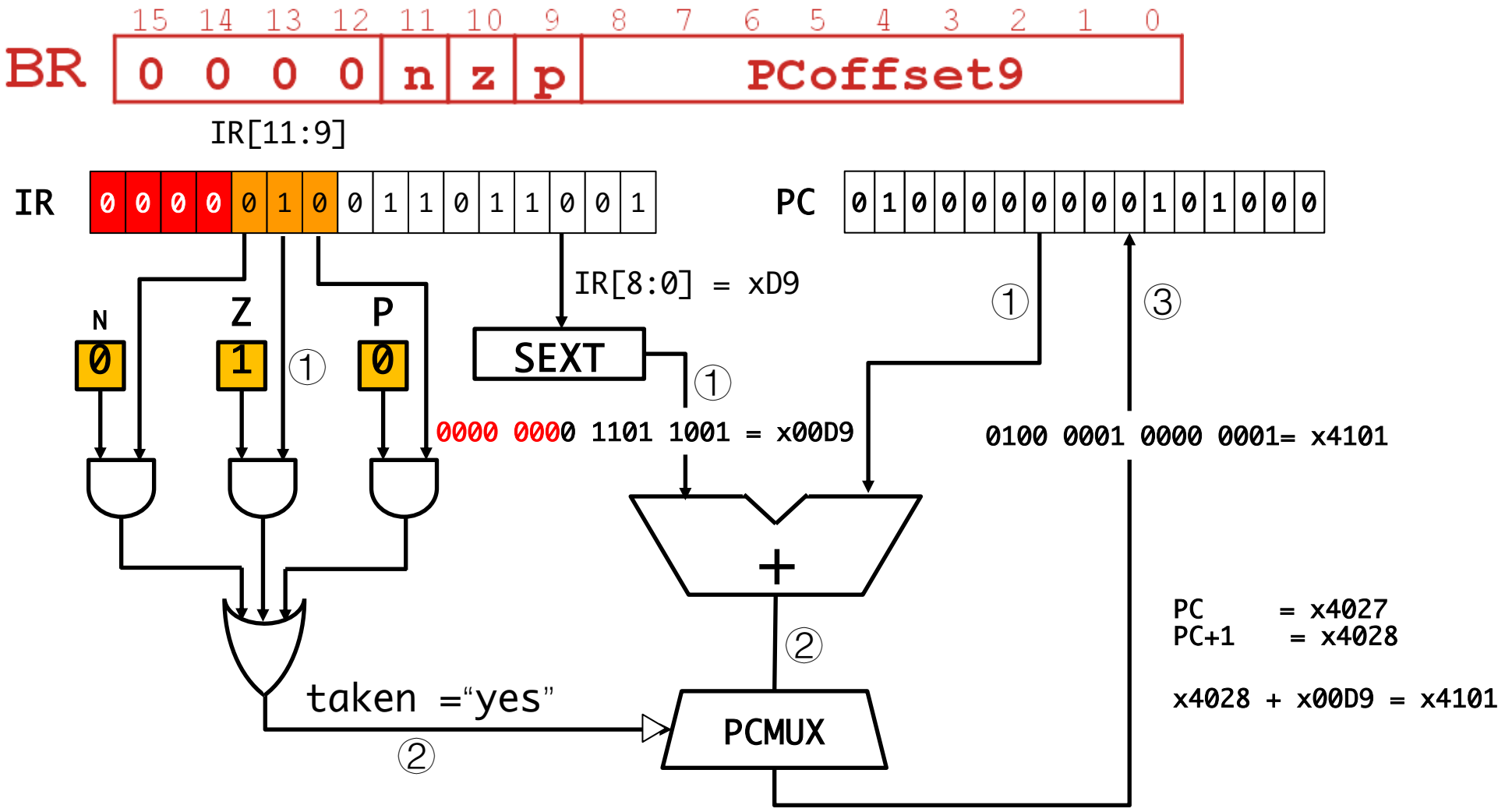
BR (PC-Relative)



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

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

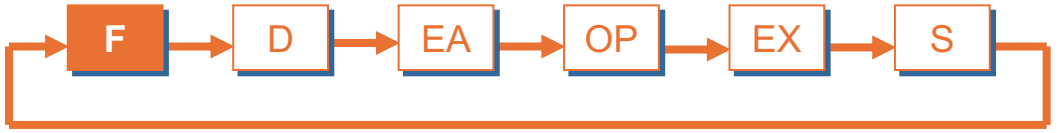
BR (PC-Relative): BR_z x4101

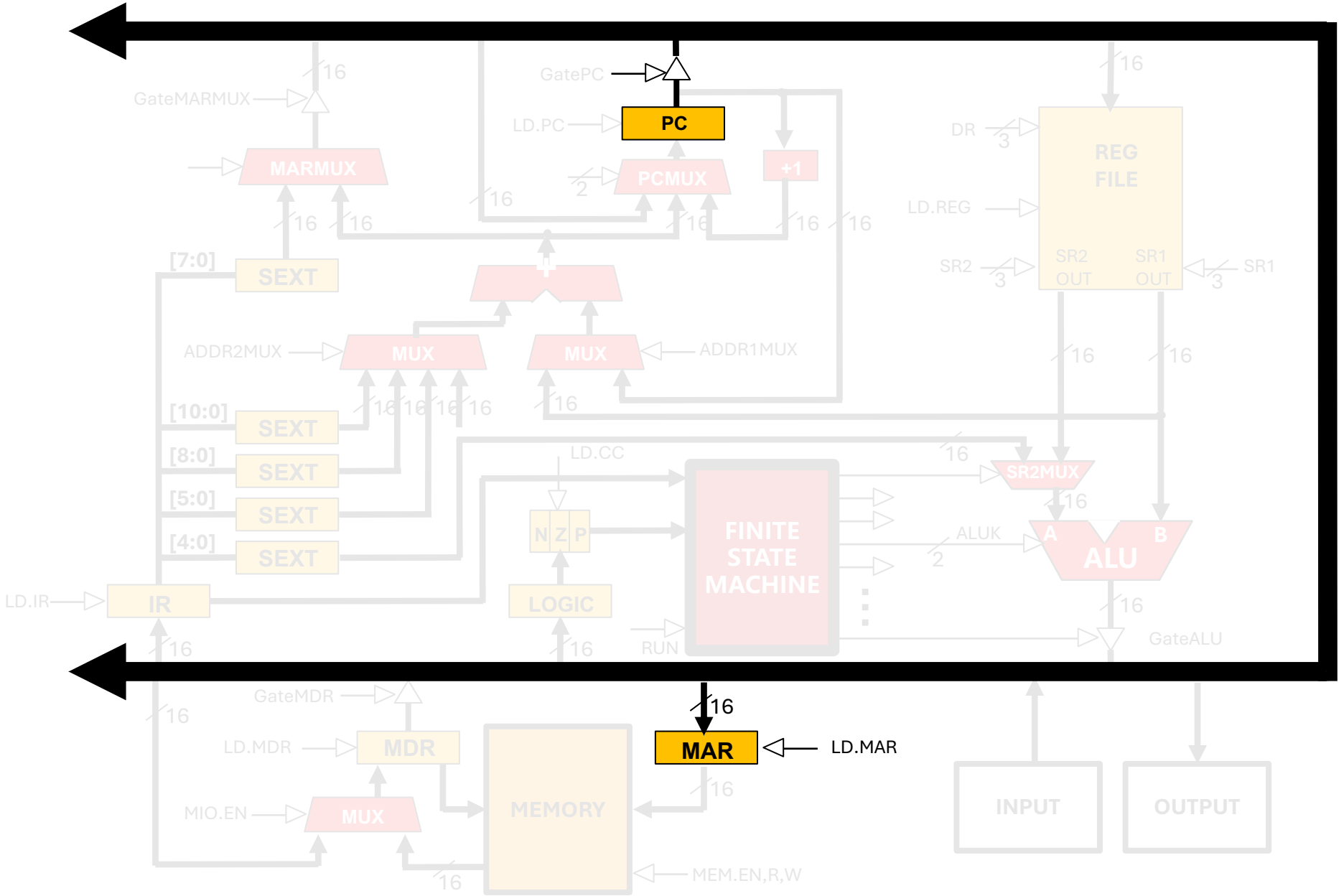


What happens if bits [11:9] are all zero?

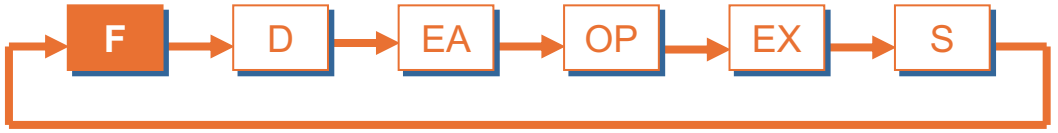
What happens if bits [11:9] are all one?

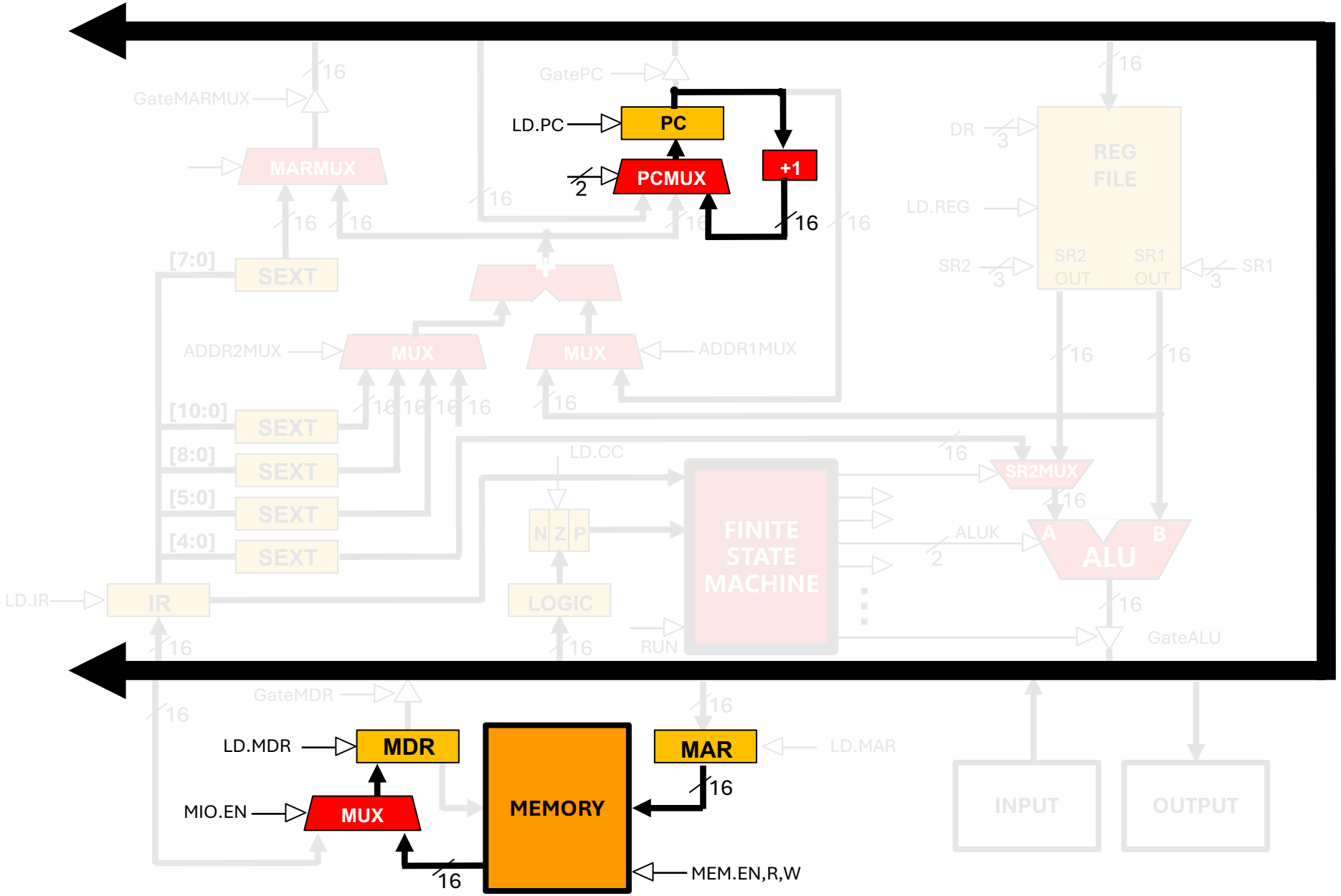
BR (PC-Relative)



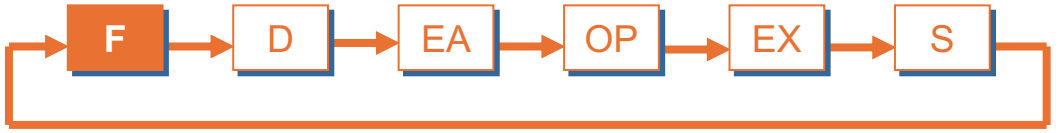


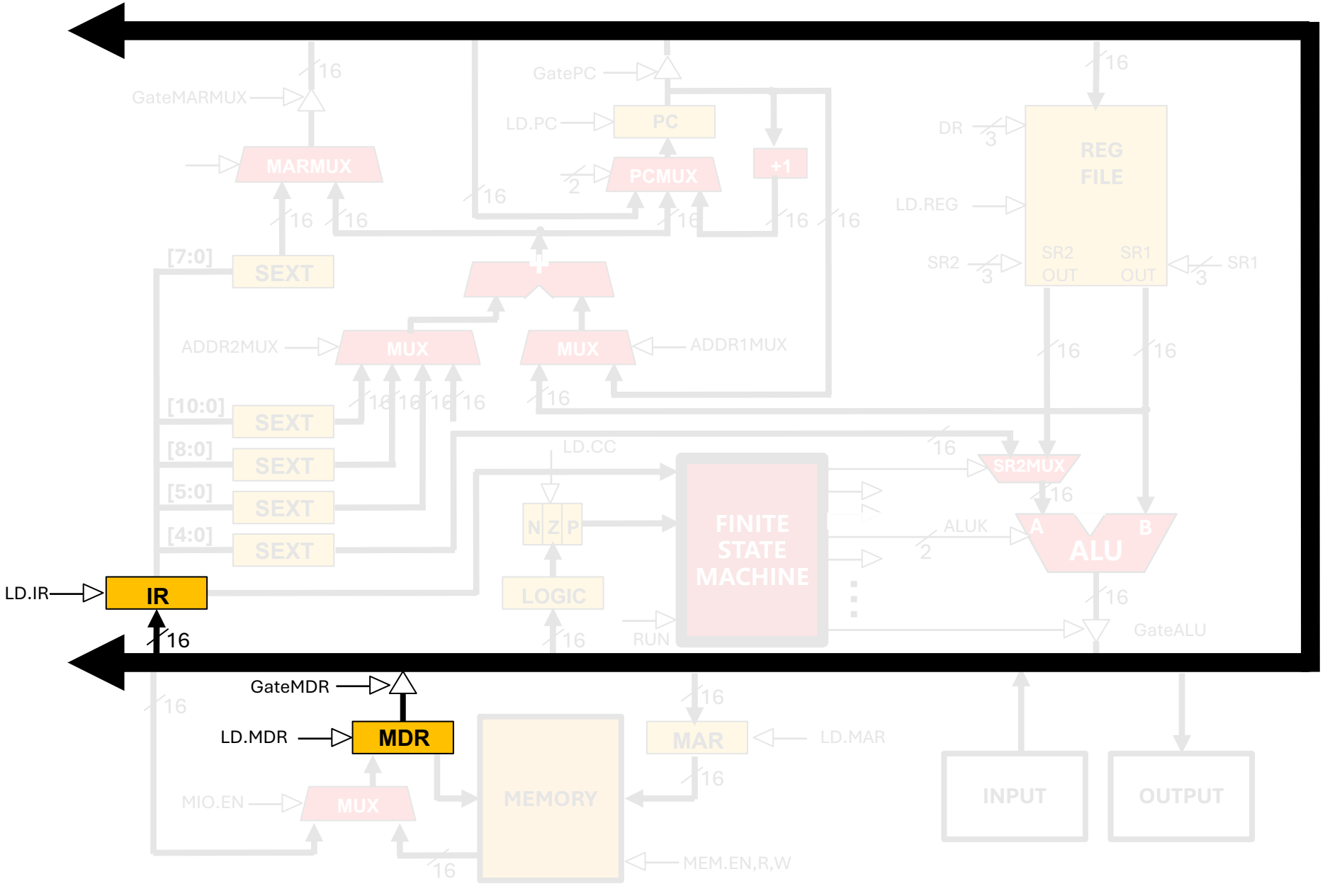
BR (PC-Relative)



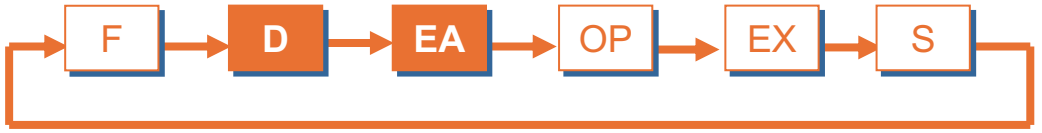


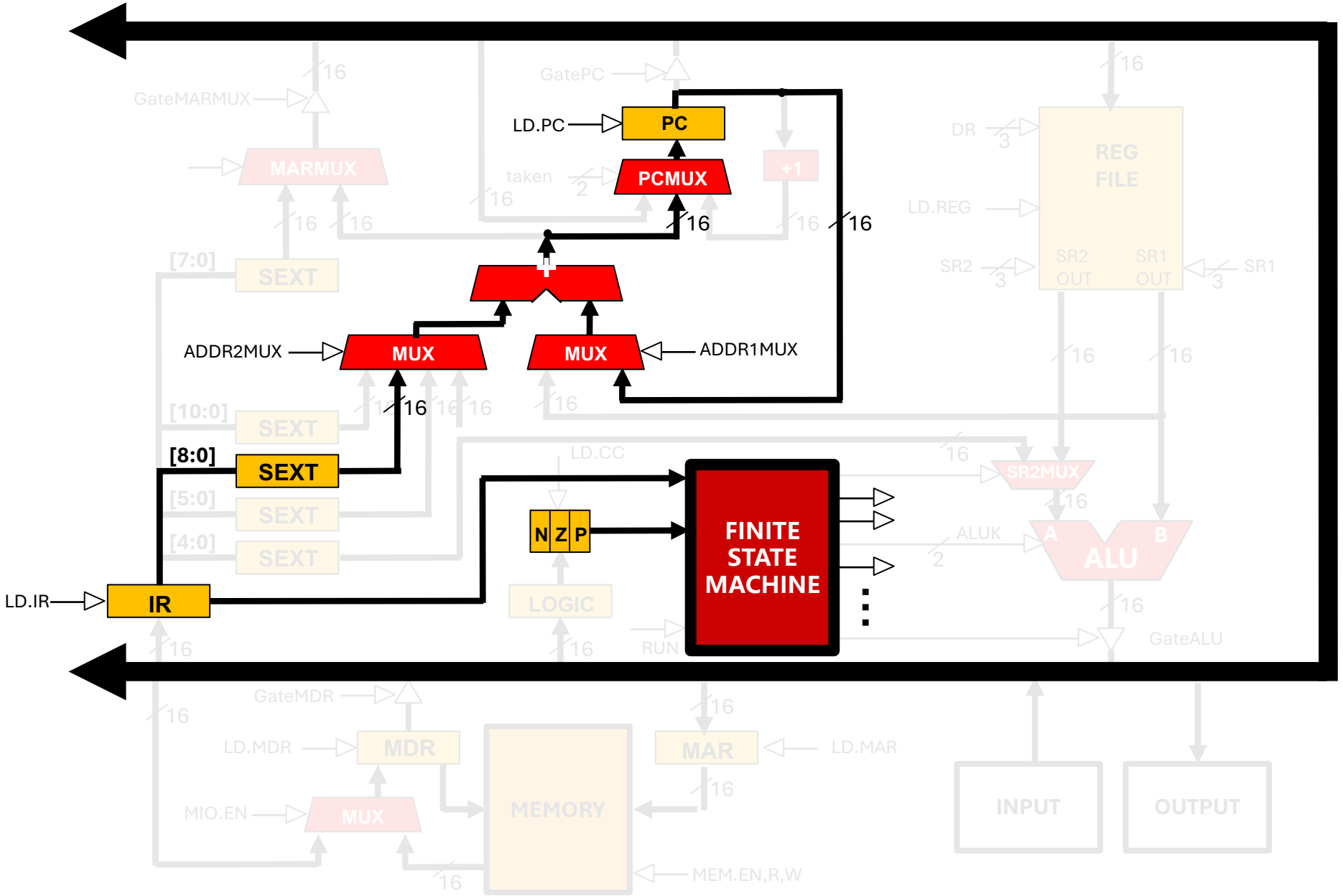
BR (PC-Relative)



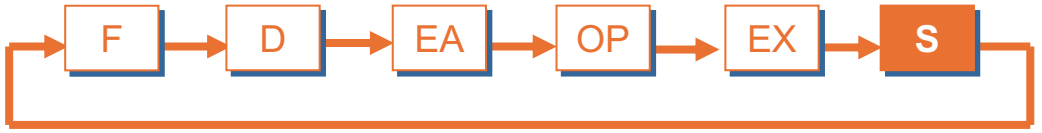


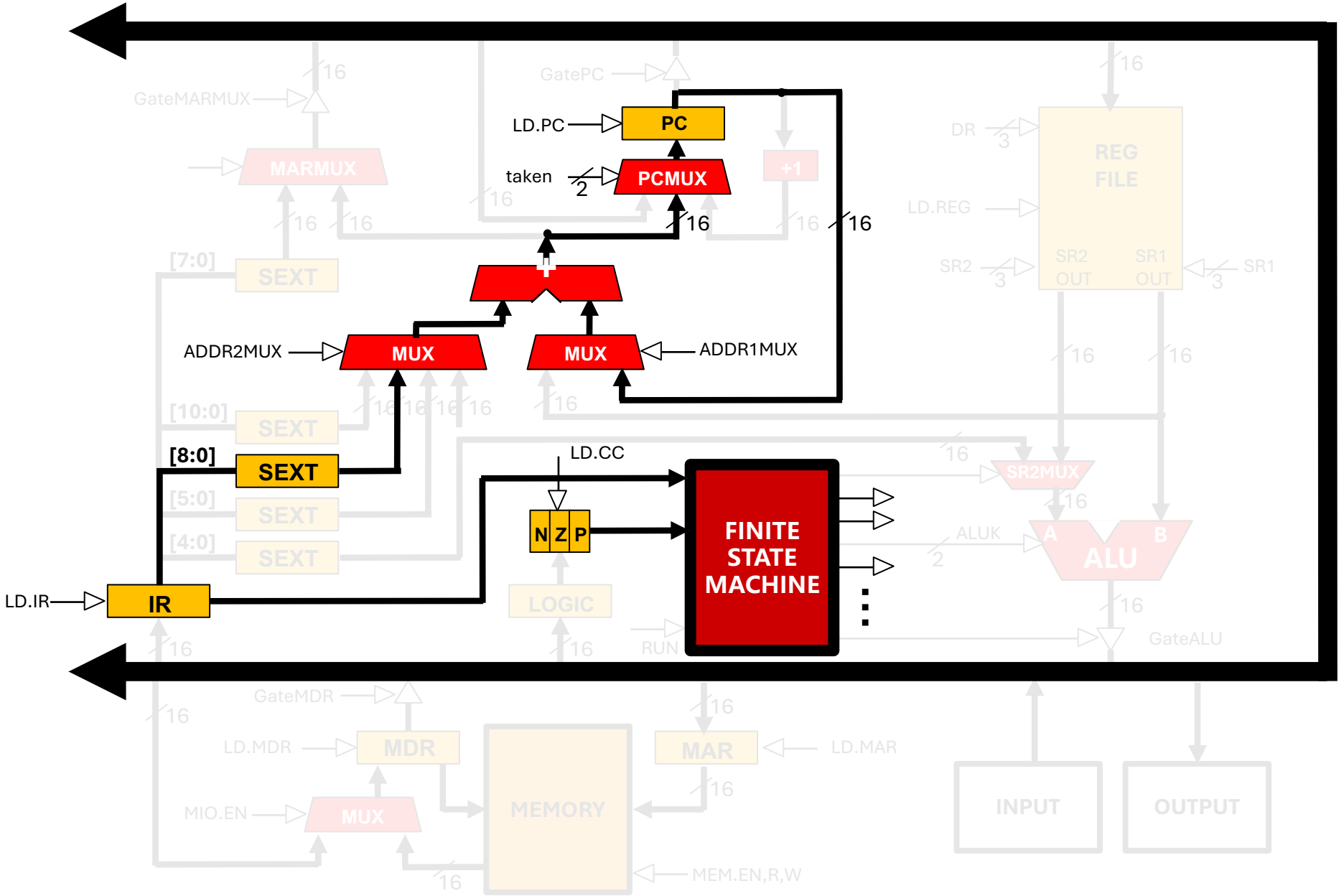
BR (PC-Relative)





BR (PC-Relative)





BR (PC-Relative)

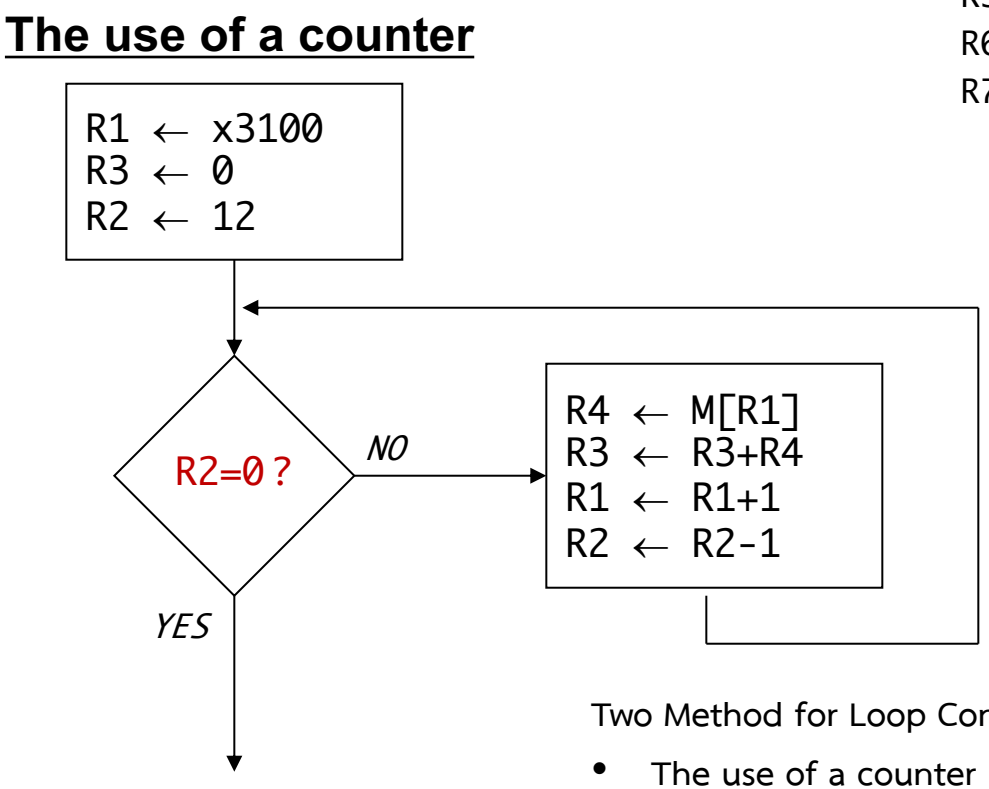
- **Check**
  - BR_{nzp} x4101 ; if (n=1 or z=1 or p=1) , JMP x4101
  - BR_{n} x4101 ; if (n=1)
  - BR_{z} x4101 ; if (z=1)
  - BR_{p} x4101 ; if (p=1)
  - BR_{nz} x4101 ; if (n=1 or z=1)
  - BR_{np} x4101 ; if (n=1 or p=1)
  - BR_{zp} x4101 ; if (z=1 or p=1)
  - BR x4101 ; PC = PC+1 → **BR x4101 == BR_{nzp} x4101**
- **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

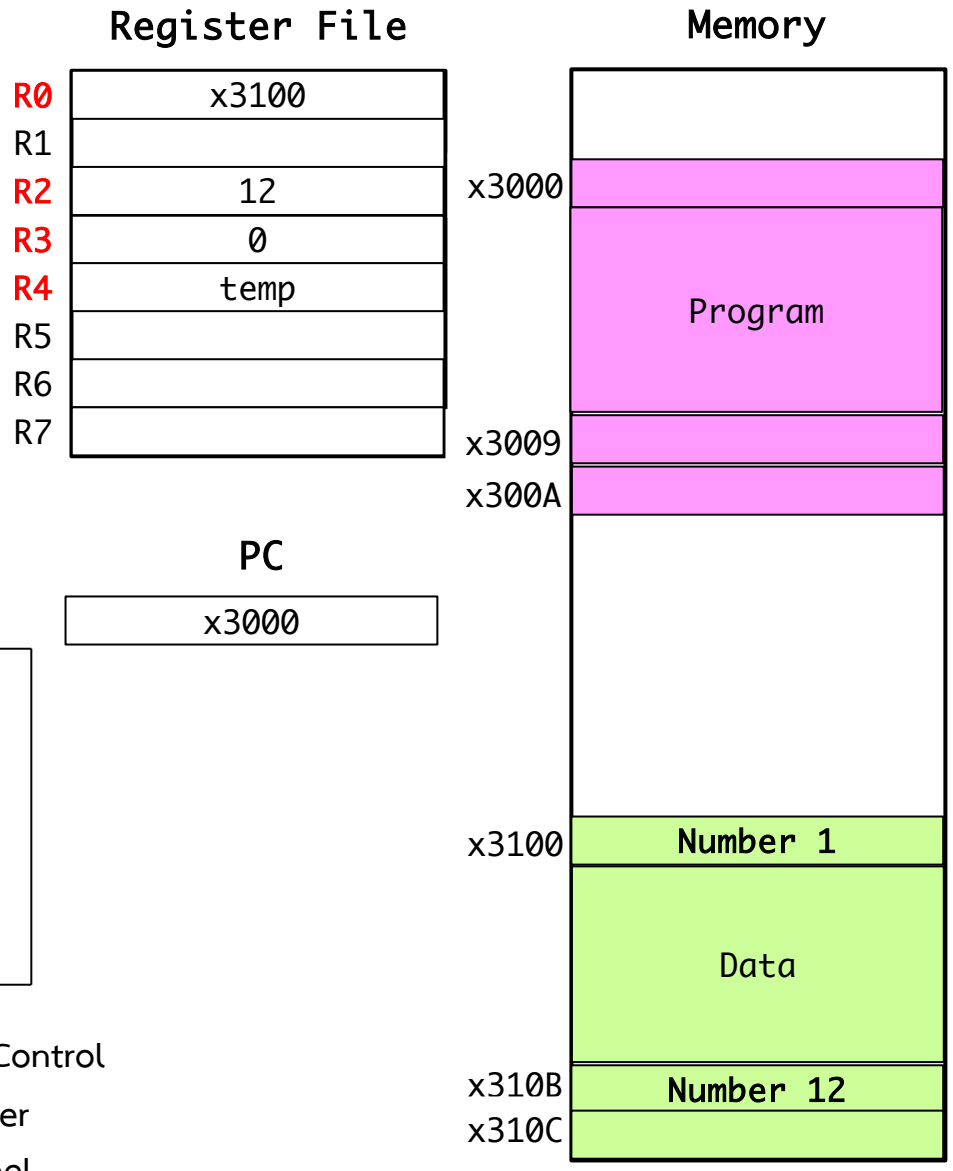
Using Branch Instructions

- Compute sum of 12 integers

  Numbers start at location x3100

  Program starts at location x3000





Two Method for Loop Control

- The use of a counter
- The use of a sentinel

Sample Summing Program (The use of a counter)

0000	BR
0001	ADD
0110	LDR
0101	AND
1110	LEA

Address	*Instruction*	*Comments*
x3000	1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1	*R1 ← x3100 (PC+0xFF); LEA*
x3001	0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0	*R3 ← 0; AND*
x3002	0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0	*R2 ← 0; AND*
x3003	0 0 0 1 0 1 0 0 1 0 1 0 1 1 0 0	*R2 ← 12; ADD*
x3004	0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1	*If Z, goto (PC+5) = x300A; BR*
x3005	0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0	*Load next value to R4; LDR*
x3006	0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0	*R3 ← R3 + R4; ADD*
x3007	0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1	*Increment R1 (pointer); ADD*
X3008	0 0 0 1 0 1 0 0 1 0 1 1 1 1 1 1	*Decrement R2 (counter); ADD*
x3009	0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0	*Goto (PC-6)=x3004 ;BR*

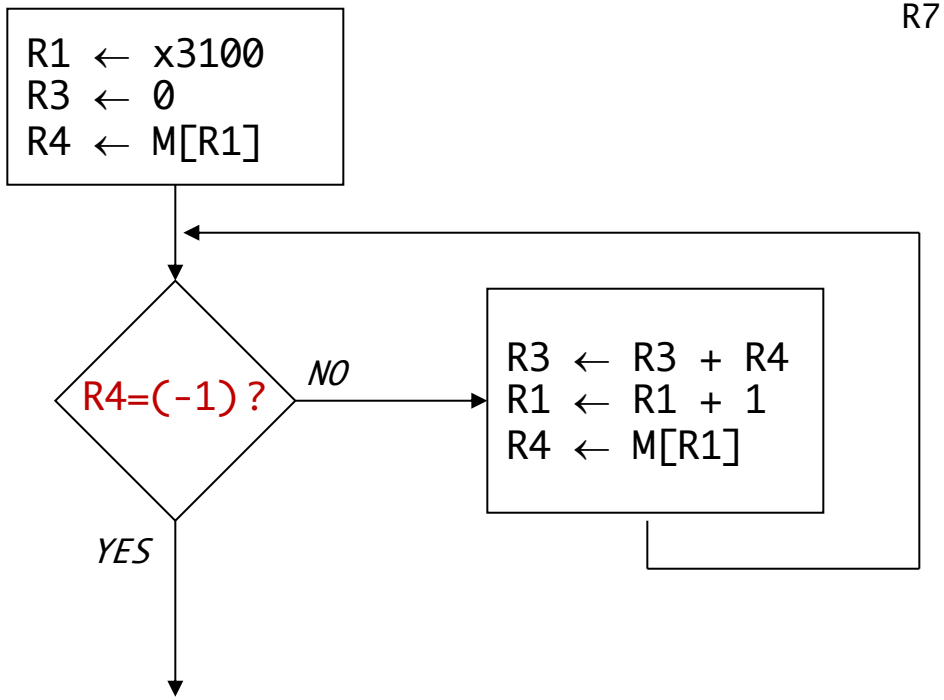
Using Branch Instructions

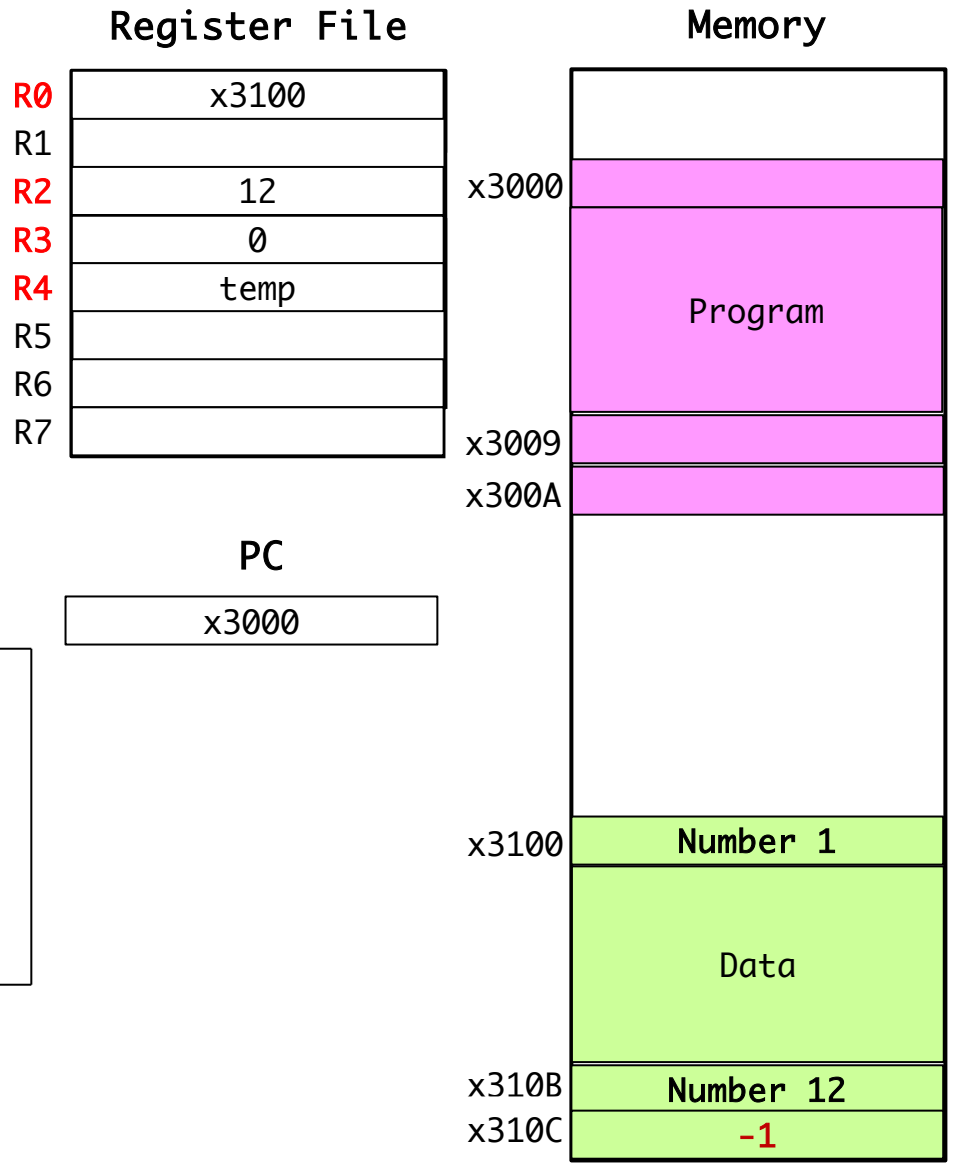
- Compute sum of 12 integers

  Numbers start at location x3100

  Program starts at location x3000

The use of a sentinel





Sample Summing Program (The use of a sentinel)

0000	BR
0001	ADD
0110	LDR
0101	AND
1110	LEA

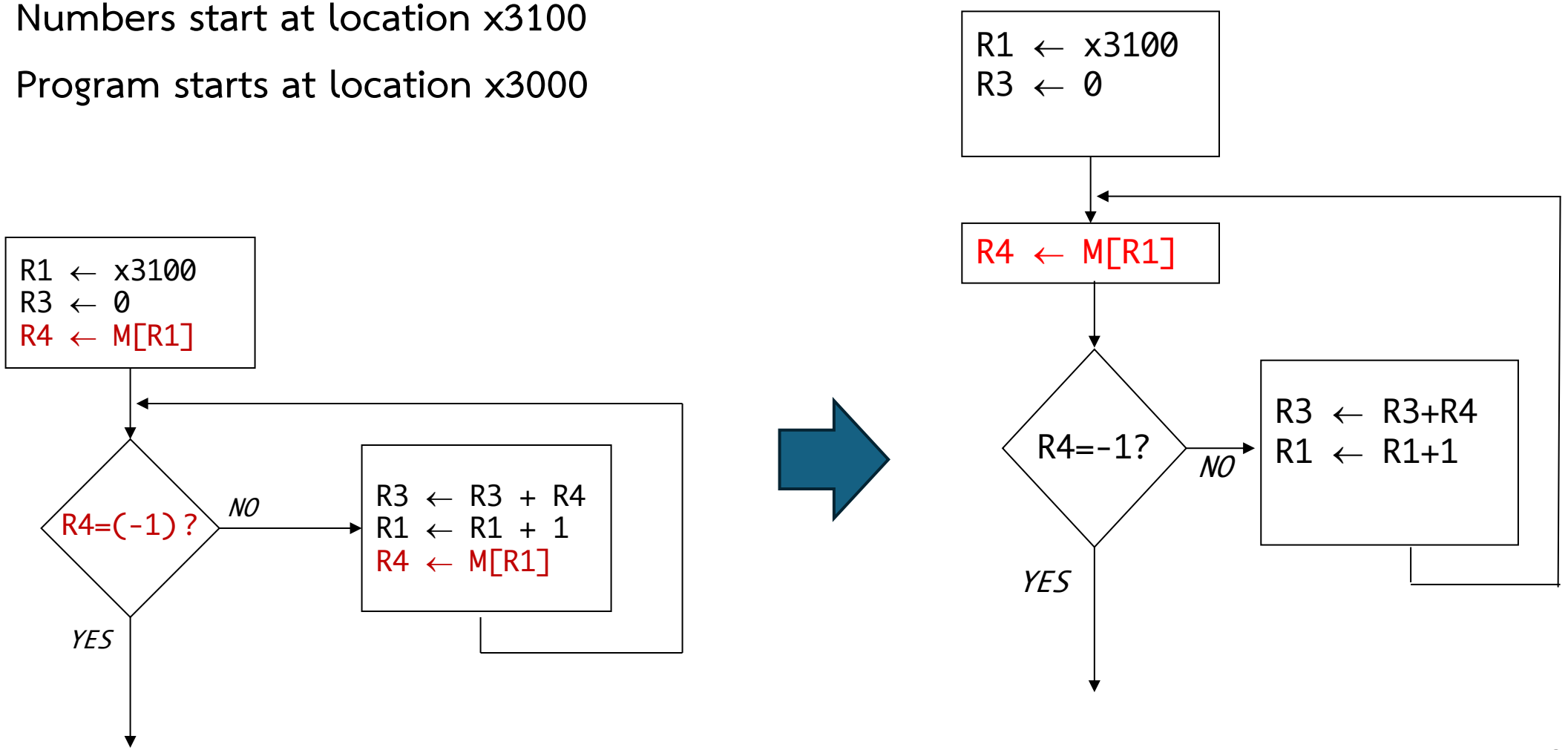
Address	*Instruction*	*Comments*
x3000	1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1	*R1 ← (PC+0xFF)=x3100 ; LEA*
x3001	0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0	*R3 ← 0; AND*
x3002	0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0	*R4 ← M[R1]; LDR*
X3003	0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0	*If N, goto (PC+ 4)=X3008; BR*
X3004	0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0	*R3 ← R3 + R4; ADD*
X3005	0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1	*R1 ← R1 + 1; ADD*
X3006	0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0	*R4 ← M[R1]; LDR*
x3007	0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1	*Goto (PC-5)= x3003; BR*

Using Branch Instructions (Code Optimization)

- Compute sum of 12 integers

  Numbers start at location x3100

  Program starts at location x3000



Sample Summing Program (Code Optimization)

0000	BR
0001	ADD
0110	LDR
0101	AND
1110	LEA

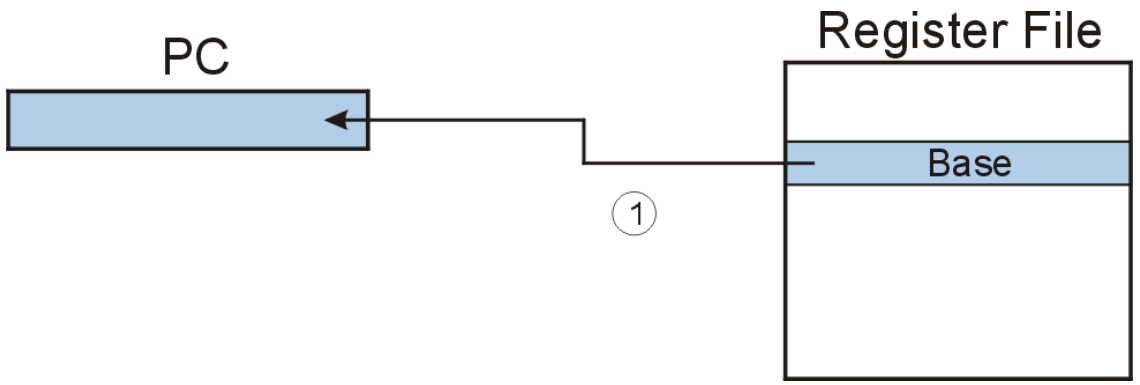
Address	*Instruction*	*Comments*
x3000	1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1	*R1 ← (PC+0xFF)= x3100*
x3001	0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0	*R3 ← 0*
x3002	0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0	*R4 ← M[R1]*
X3003	0 0 0 0 1 0 0 0 0 0 0 0 0 1 1	*If N, goto (PC+ 3)=X3007*
X3004	0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0	*R3 ← R3 + R4*
X3005	0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1	*R1 ← R1 + 1*
x3006	0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1	*Goto (PC-5)= x3002*

LC-3 Jump & TRAP Instruction

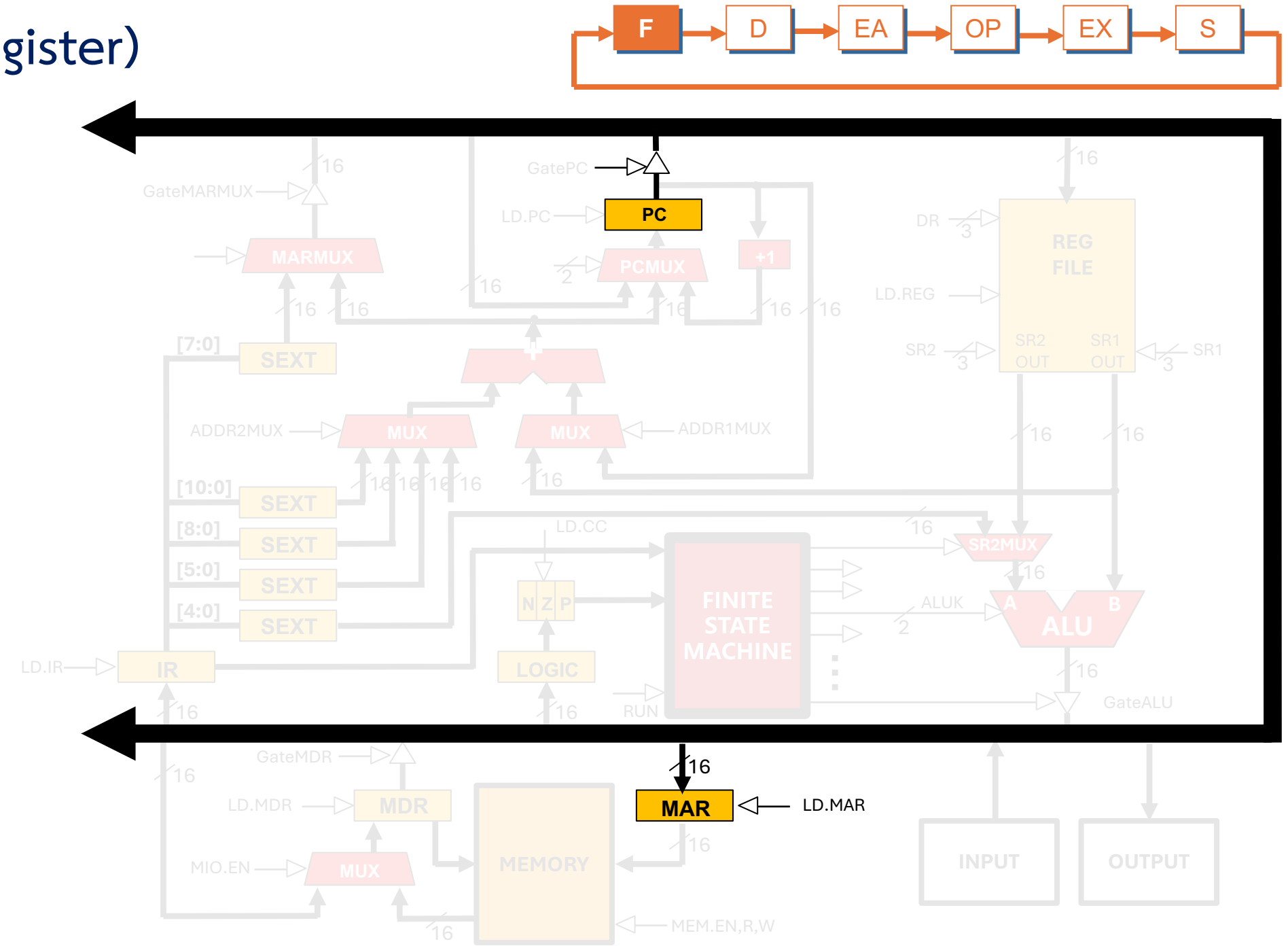
JMP (Register)

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

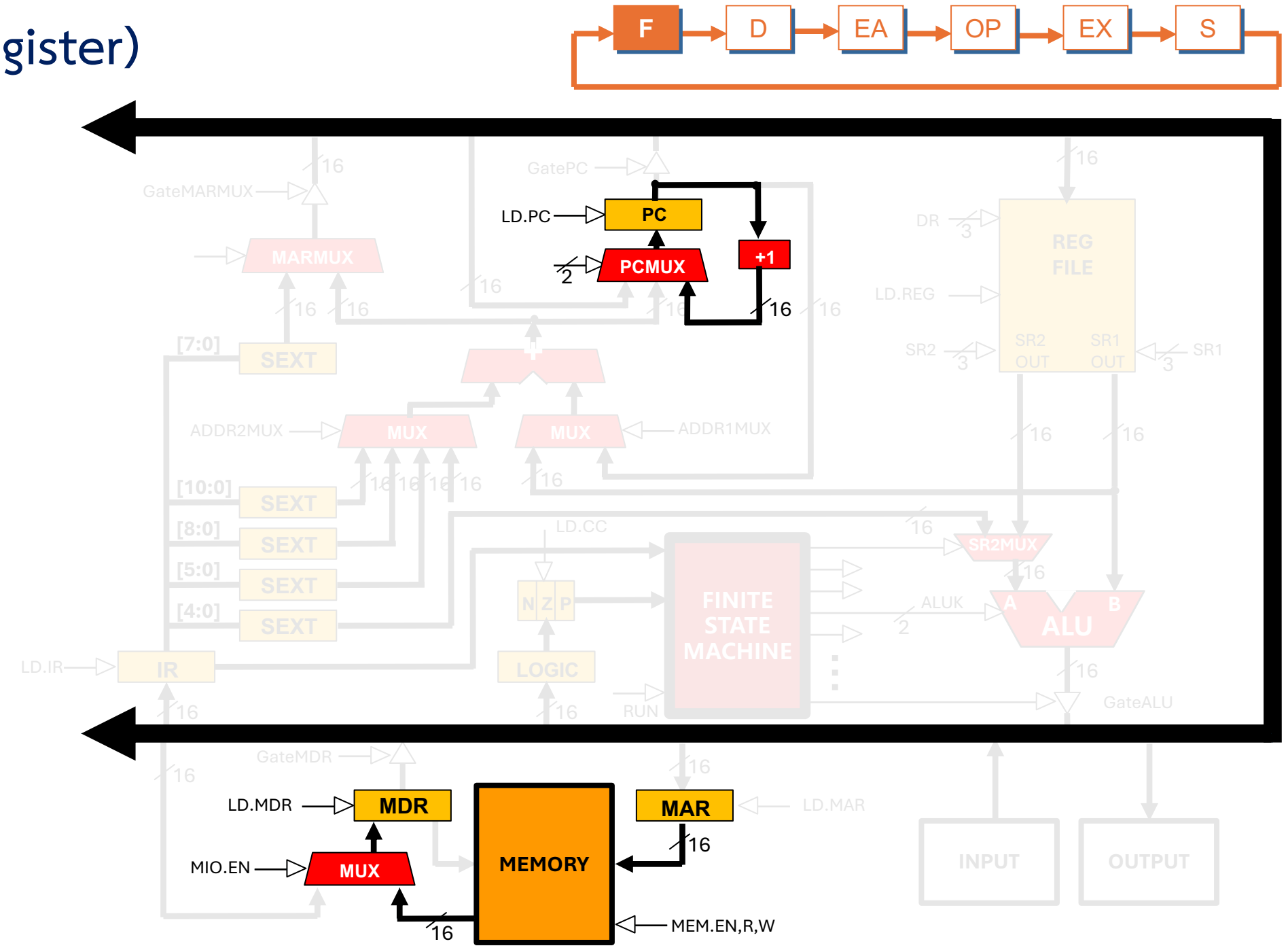
- Target address is the contents of a register.
- Allows any target address.



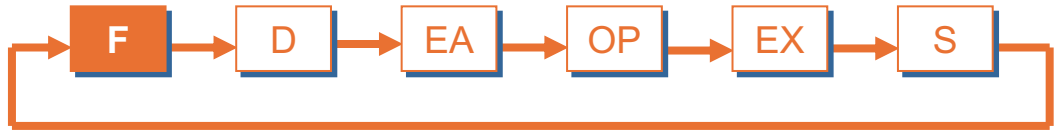
JMP (Register)

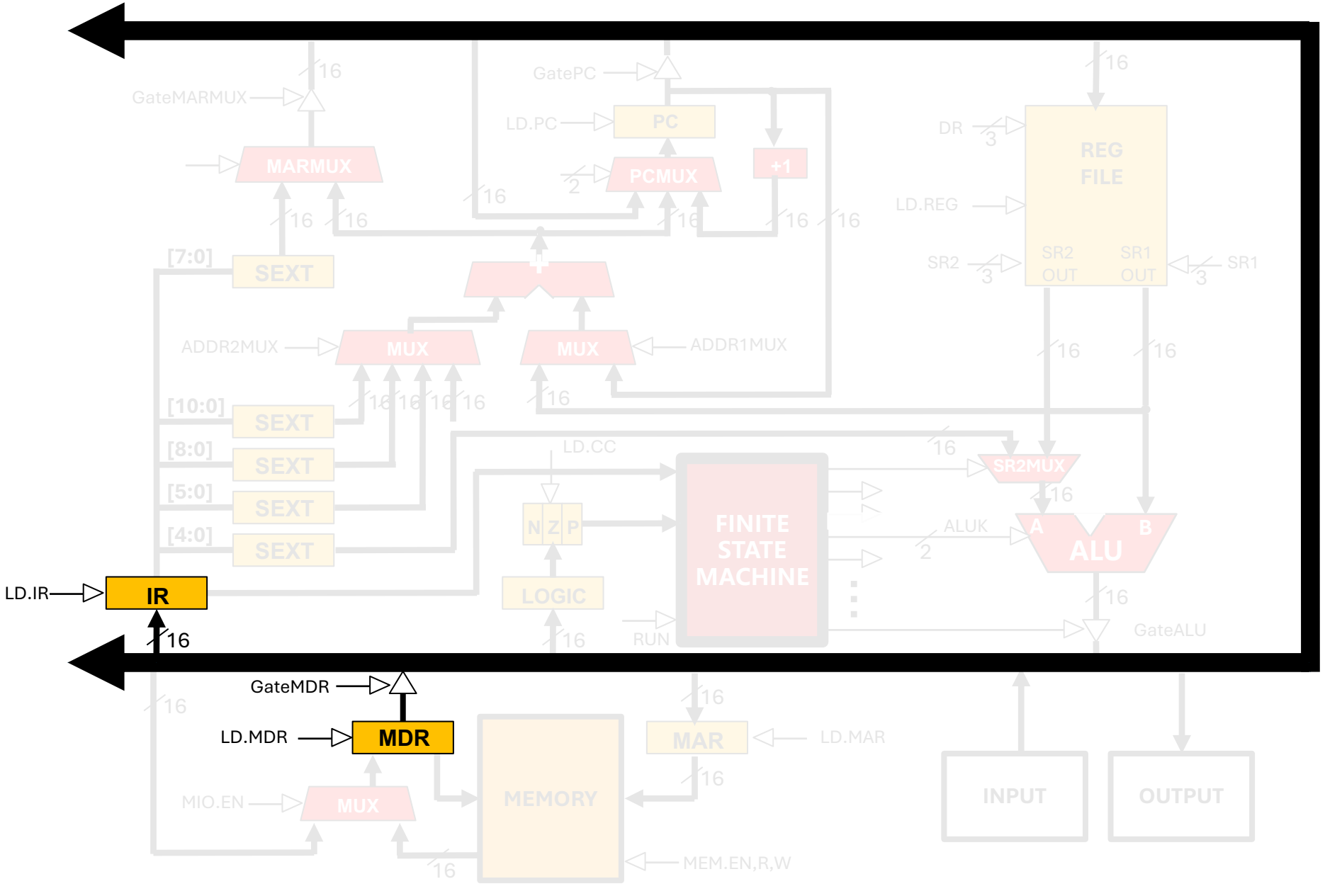


JMP (Register)

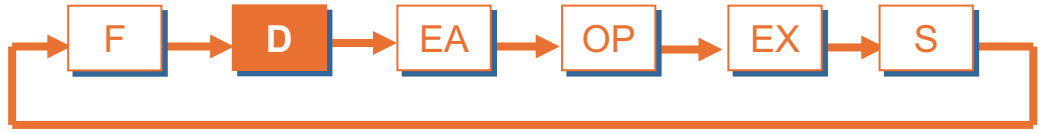


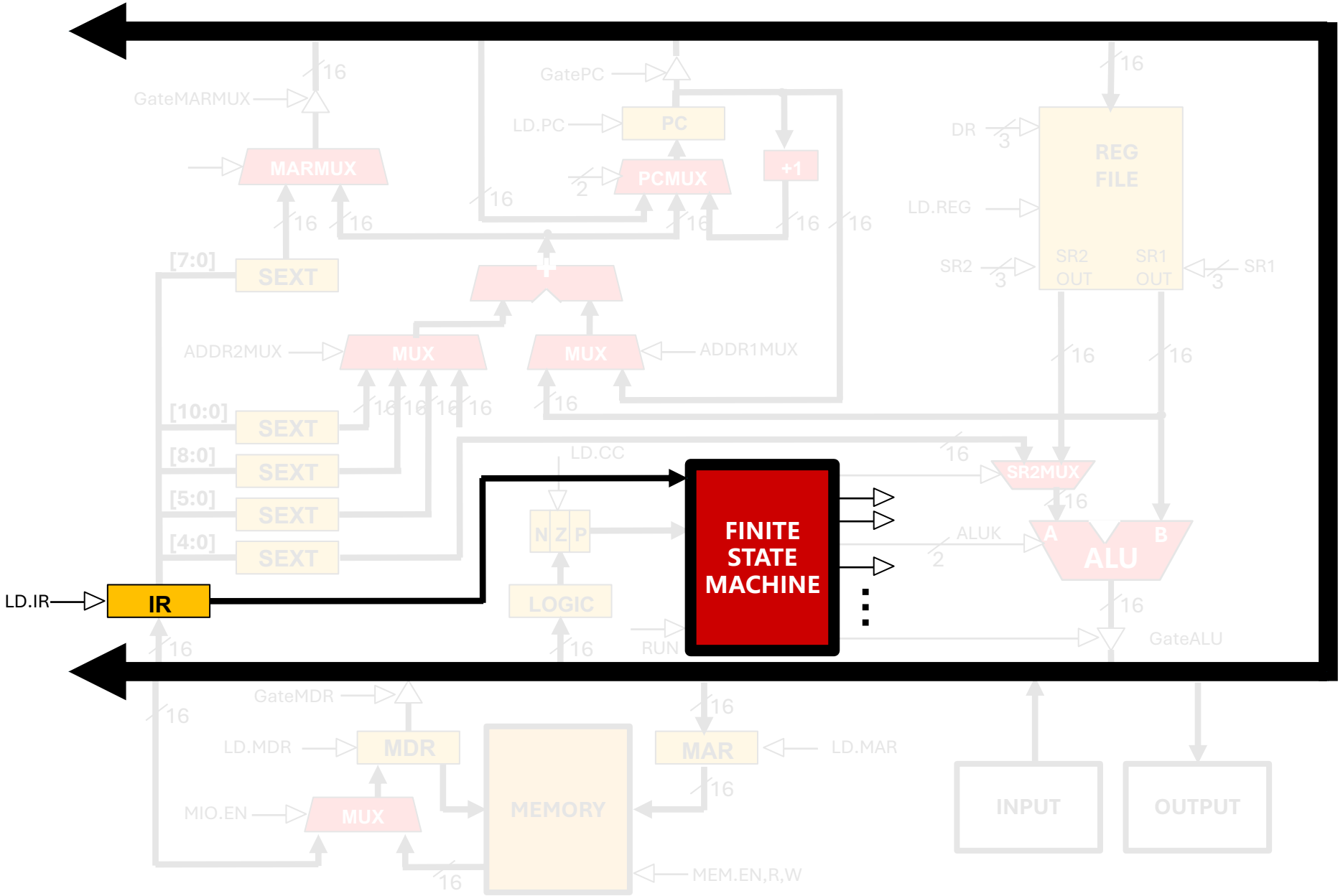
JMP (Register)



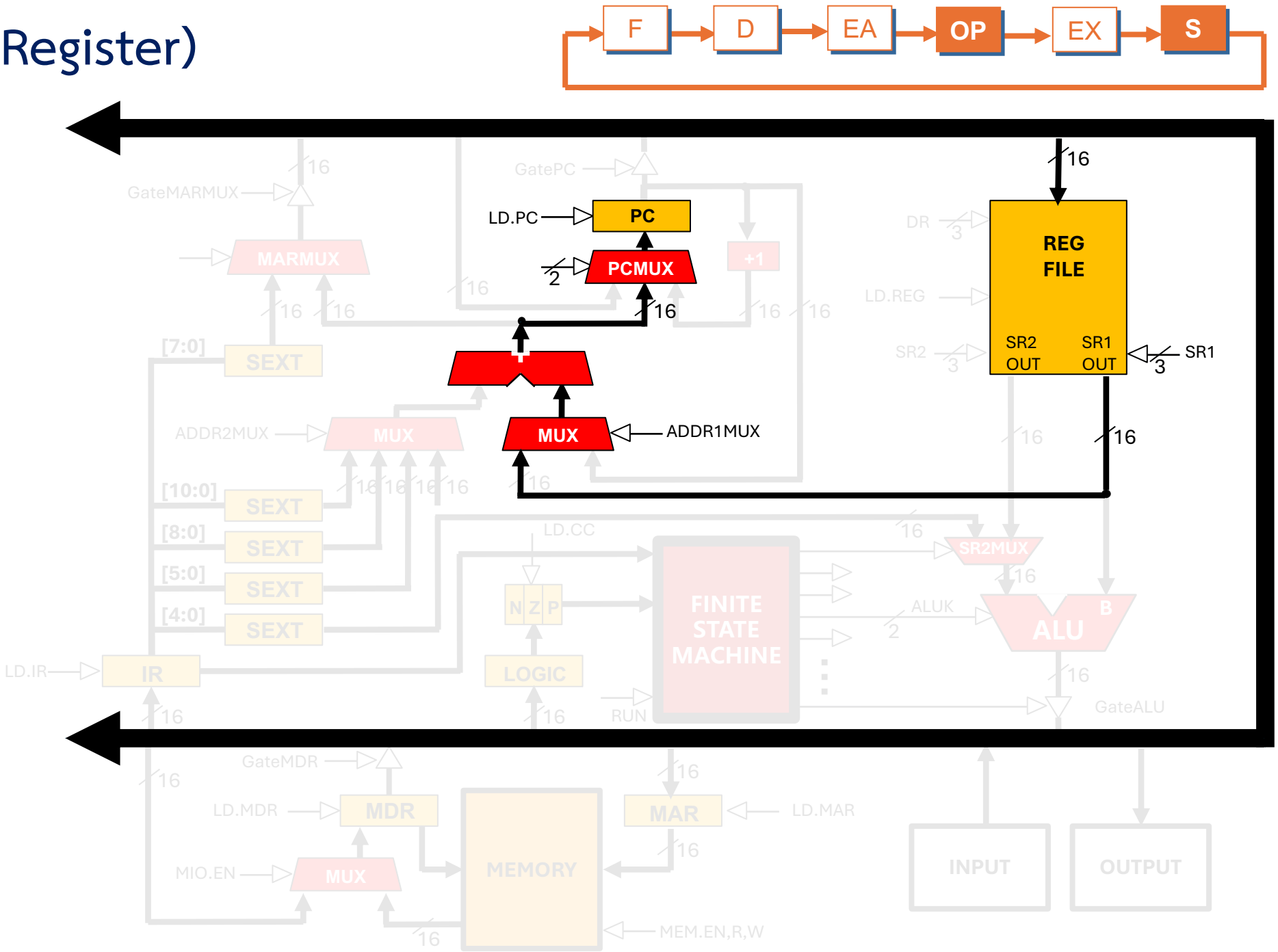


JMP R7 (Register)





JMP R7 (Register)



TRAP

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.

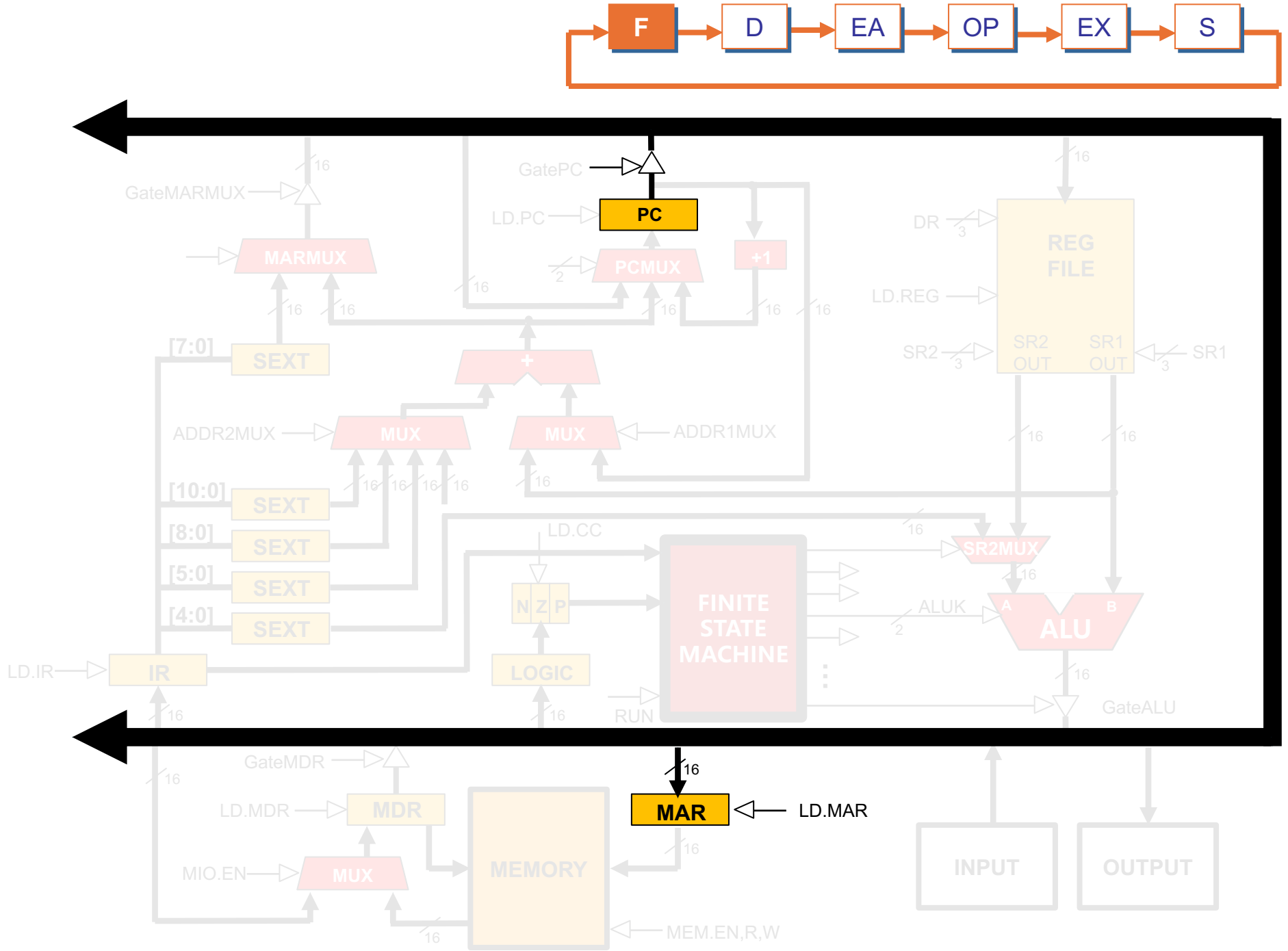
TRAP

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

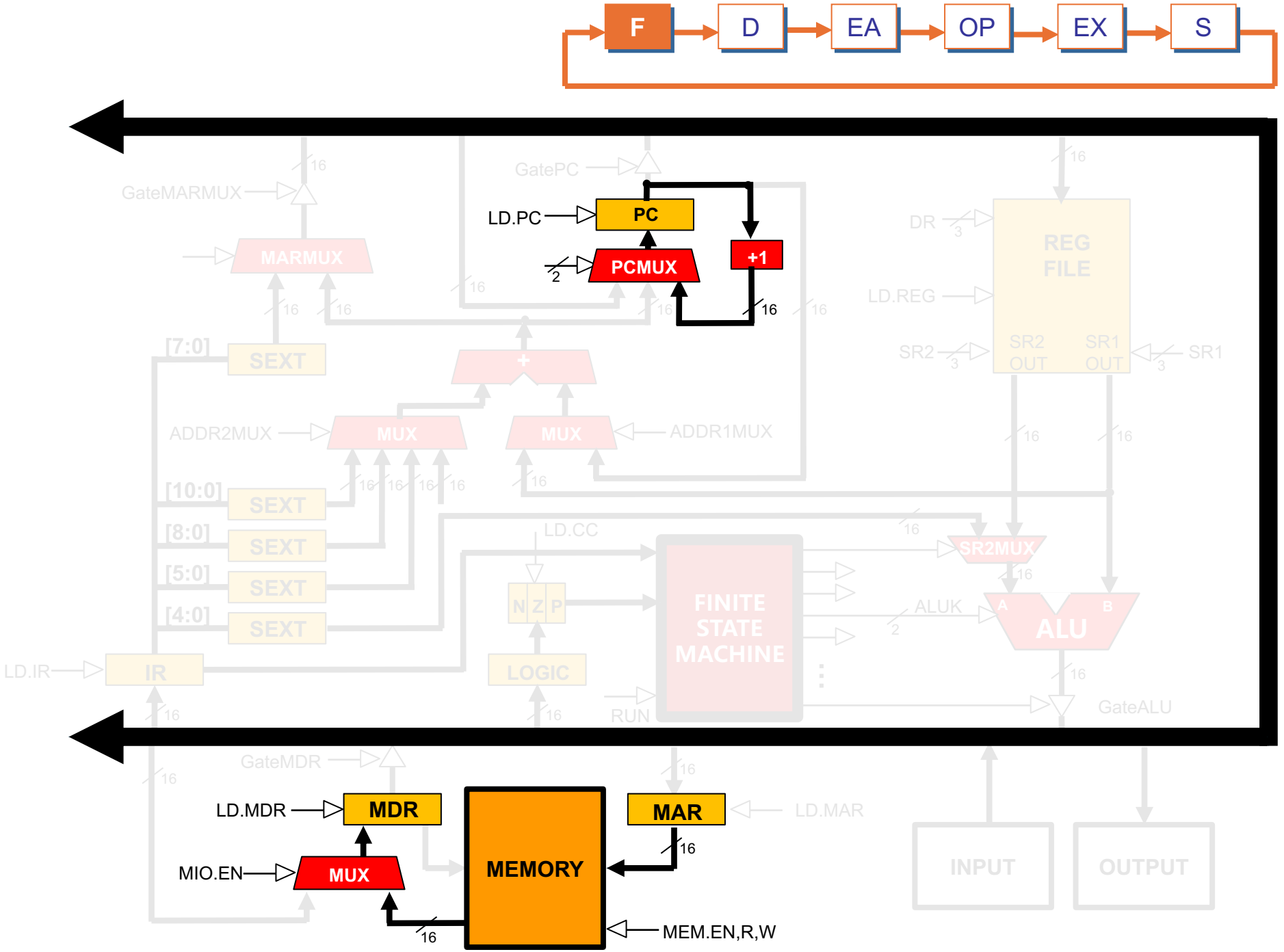
vector	*routine*
x23	input a character from the keyboard
x21	output a character to the monitor
x25	halt the program

When routine is done, PC is set to the instruction following TRAP

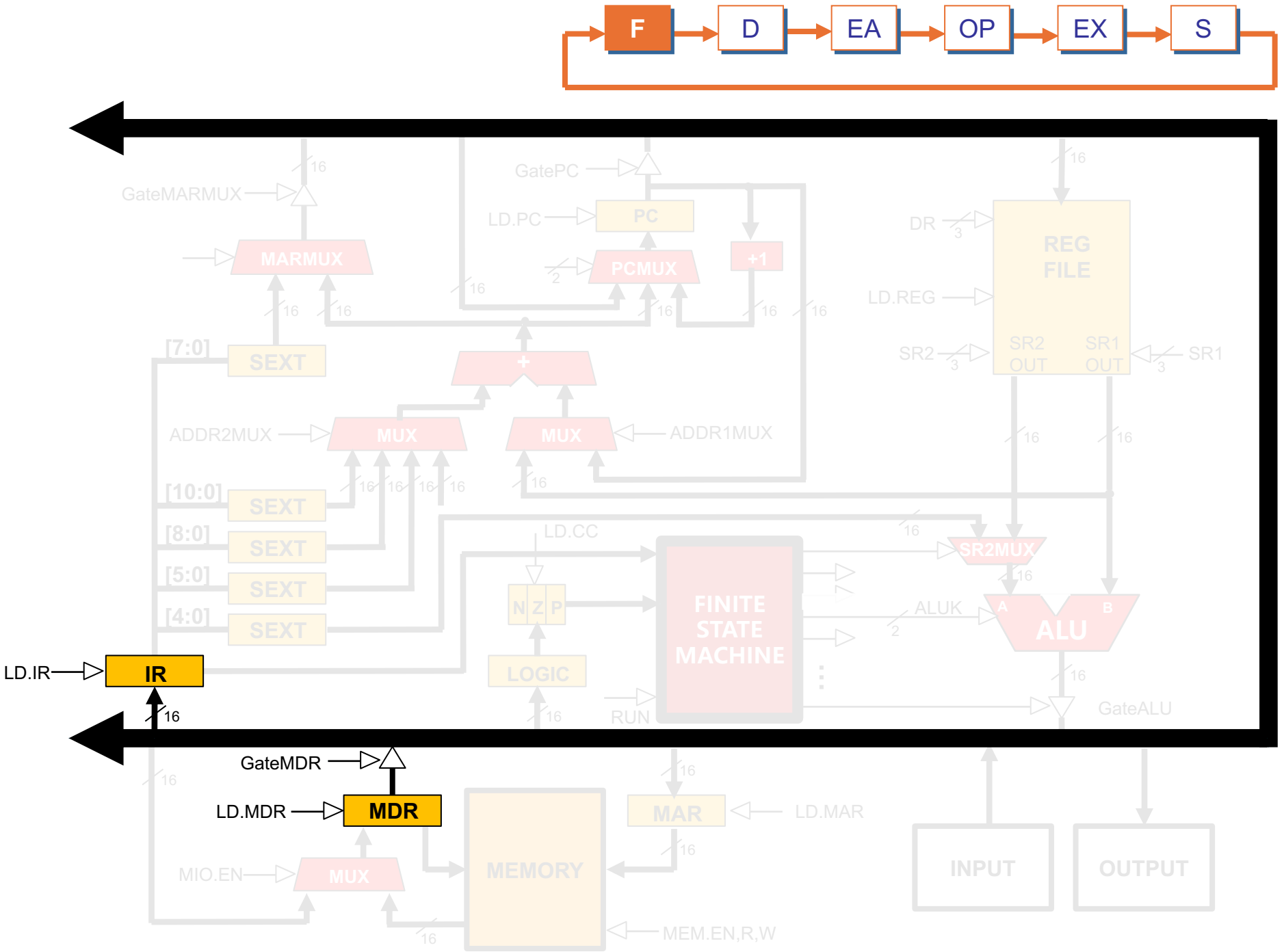
TRAP


F
D
EA
OP
EX
S
GatePC
LD.PC
PC
GateMARMUX
MARMUX
PCMUX
+1
+
DR
REG FILE
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
[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
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MIO.EN
MUX
MEMORY
MAR
LD.MAR
INPUT
OUTPUT
MEM.EN,R,W

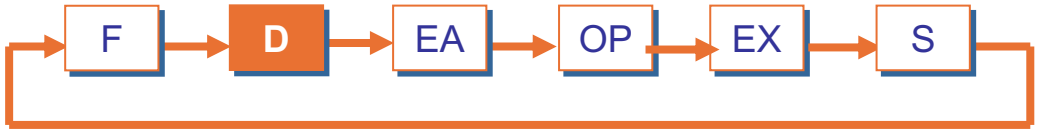
TRAP

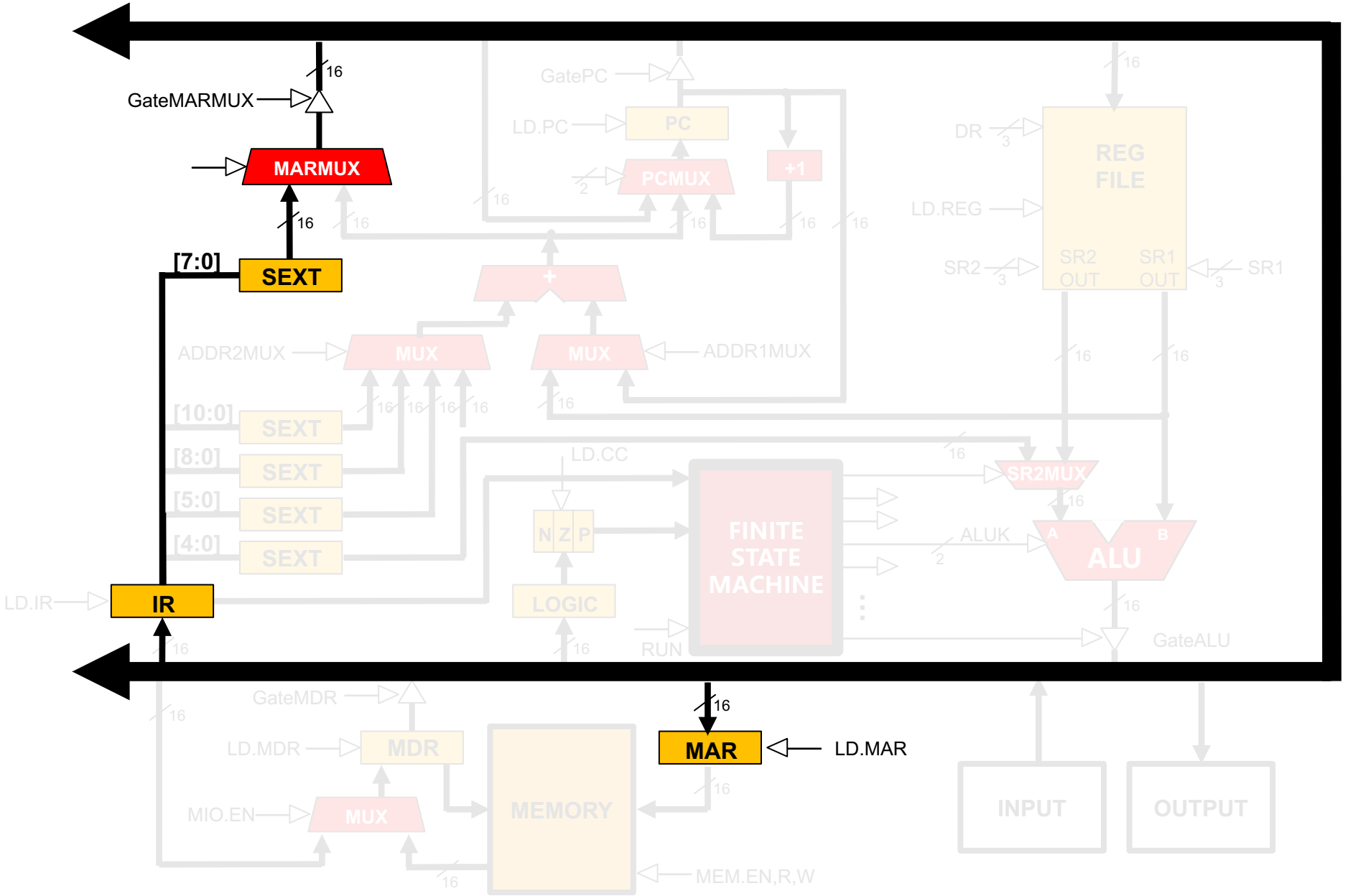

F
D
EA
OP
EX
S
GatePC
LD.PC
PC
PCMUX
+1
16
2
GateMARMUX
MARMUX
16
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
SEXT
[8:0]
SEXT
[5:0]
SEXT
[4:0]
SEXT
LD.CC
N Z P
FINITE STATE MACHINE
RUN
LD.IR
IR
LOGIC
DR
3
REG FILE
LD.REG
SR2
SR2 OUT
SR1 OUT
SR1
SR2MUX
ALUK
A
B
ALU
GateALU
GateMDR
LD.MDR
MDR
MIO.EN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

TRAP

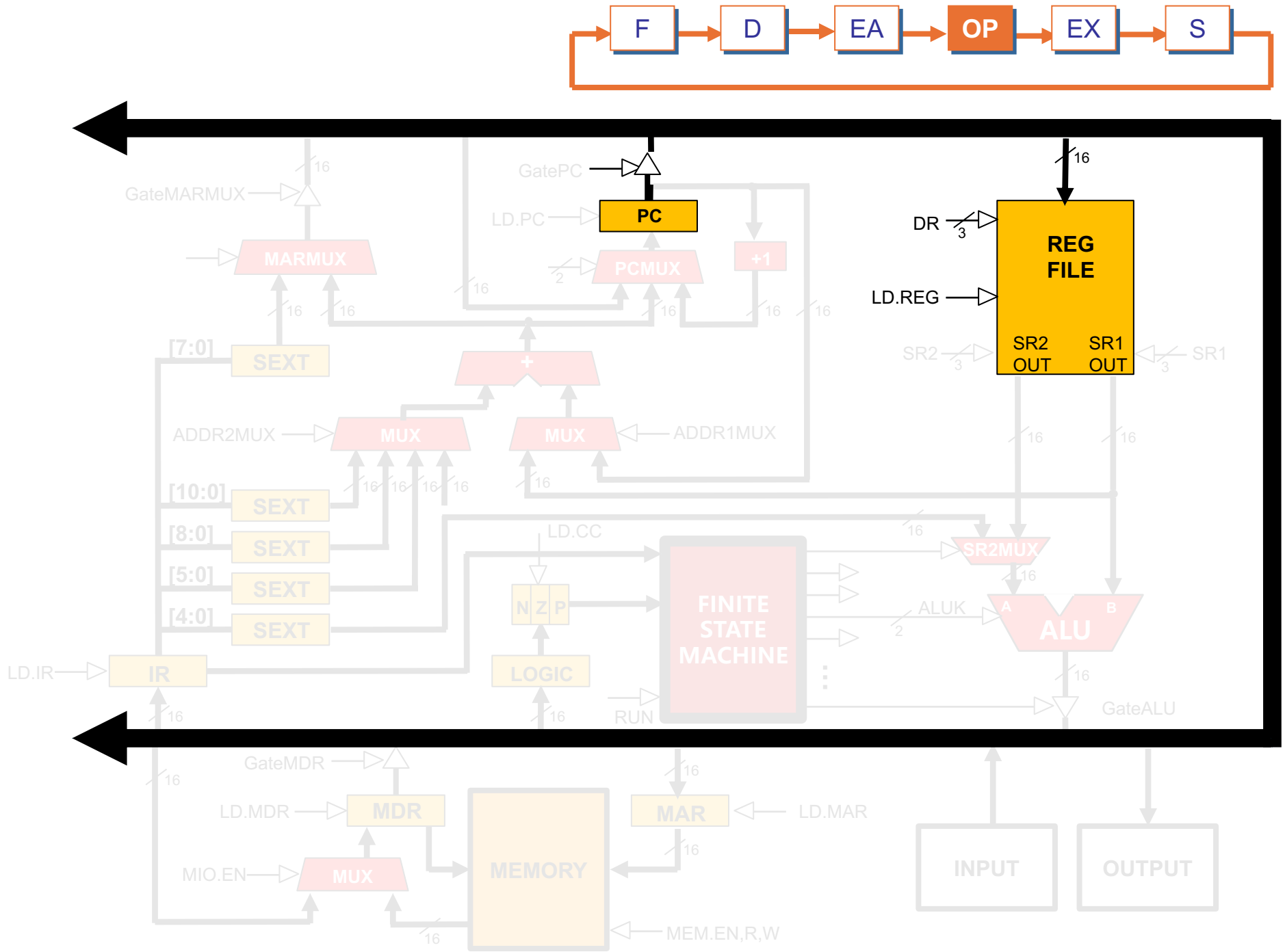

F
D
EA
OP
EX
S
GateMARMUX
GatePC
LD.PC
PC
MARMUX
PCMUX
+1
DR
REG FILE
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
[7:0]
SEXT
+
ADDR2MUX
MUX
MUX
ADDR1MUX
[10:0]
[8:0]
[5:0]
[4:0]
LD.CC
SR2MUX
N Z P
FINITE STATE MACHINE
ALUK
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MAR
LD.MAR
MIO.EN
MUX
MEMORY
INPUT
OUTPUT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

TRAP

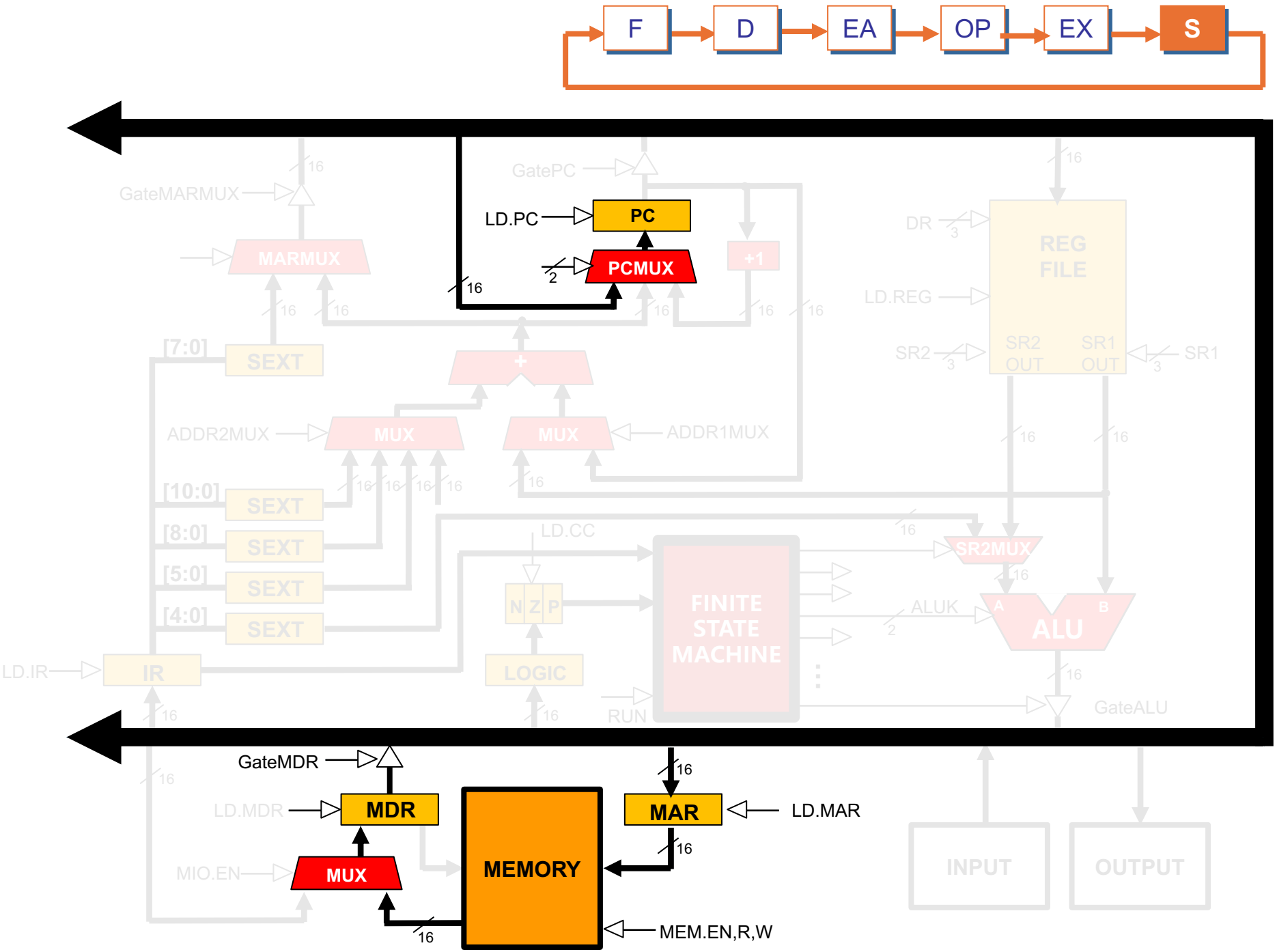

F
D
EA
OP
EX
S


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

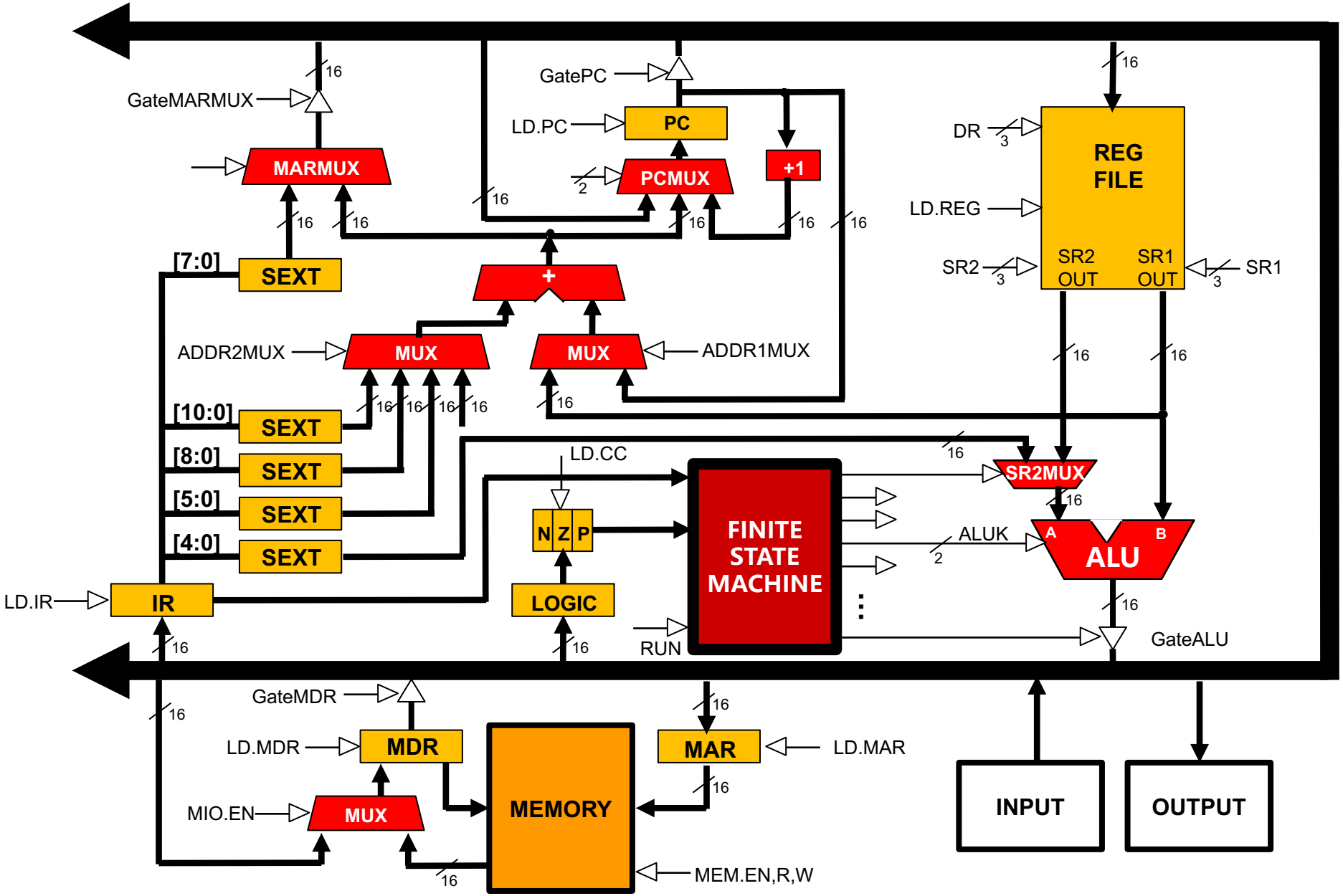
TRAP


F
D
EA
OP
EX
S
GatePC
LD.PC
PC
+1
PCMUX
GateMARMUX
MARMUX
16
DR
3
REG FILE
LD.REG
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
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MAR
LD.MAR
MIO.EN
MUX
MEMORY
INPUT
OUTPUT
MEM.EN,R,W

TRAP


F
D
EA
OP
EX
S
GatePC
LD.PC
PC
PCMUX
+1
GateMARMUX
MARMUX
DR
REG FILE
LD.REG
SR2
SR1
SR2 OUT
SR1 OUT
[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
A
B
ALU
LD.IR
IR
LOGIC
RUN
GateALU
GateMDR
LD.MDR
MDR
MAR
LD.MAR
MIO.EN
MUX
MEMORY
MEM.EN,R,W
INPUT
OUTPUT
16
2
3

LC-3 Data Path After Control Instruction



Q & A