ECE290 Fall 2012
Lecture 21

Dr. Zbigniew Kalbarczyk

Today

- LC -3 Instruction Cycle
- LC-3 State Diagram
- Location of Control Signals in LC-3
- Values of Control Signals in LC-3 for ADD operation
- Values of Control Signals in LC-3 for NOT operation

Instruction Cycle

Phases of Instruction Cycle:

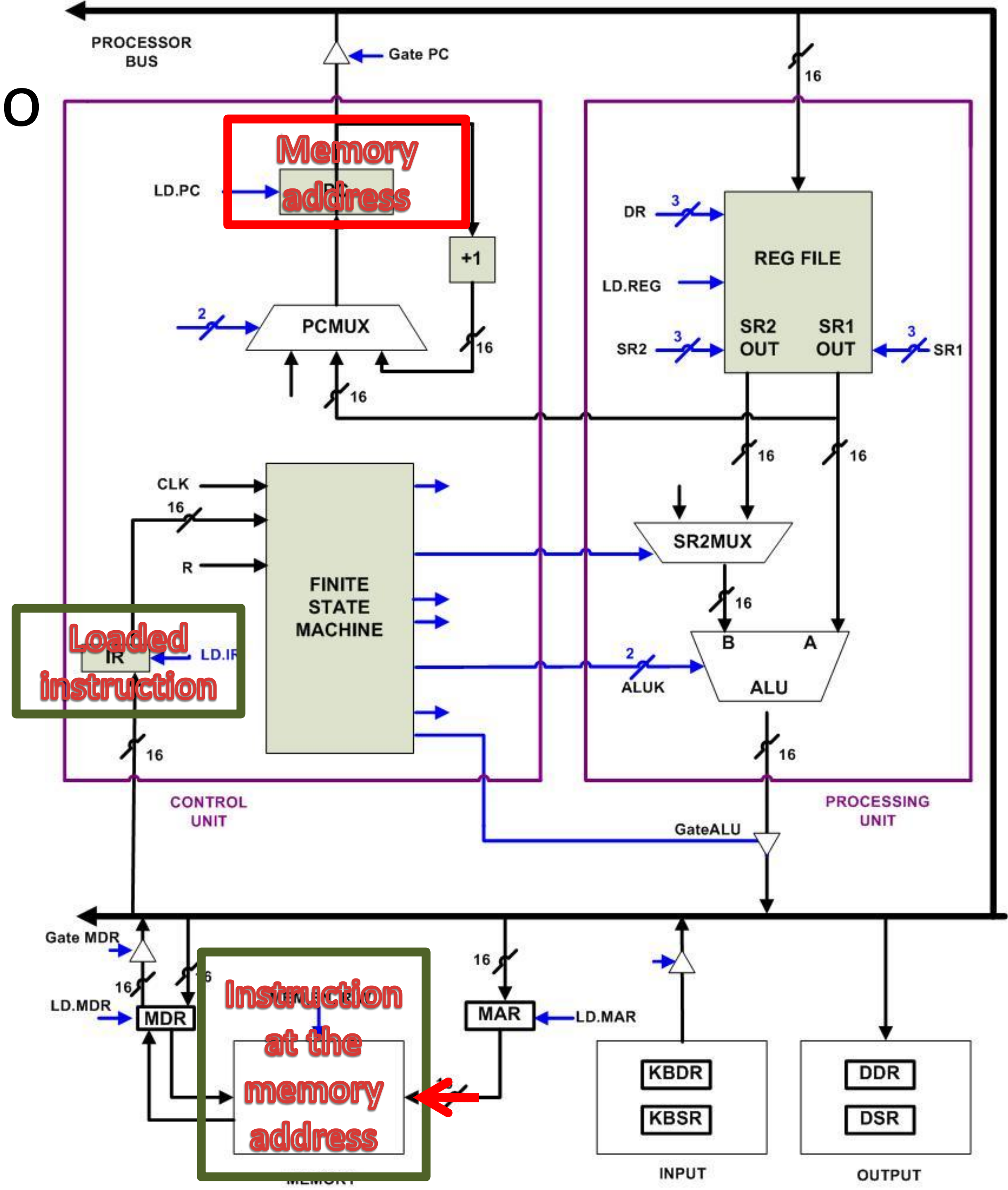
- **Fetch Phase:**
  - Fetch word from memory into instruction register (IR)
- **Decode Phase:**
  - Decode word in IR as instruction
- **Execute Phase**
  - Execute instruction

Instruction Cycle: Fetch Phase

- **Fetch Phase:** fetch word from memory into IR
  - Place memory address in PC on the bus and load MAR with the memory address
  - Increment PC
  - Read from memory address specified in MAR to MDR
  - Copy MDR into IR

Instructio

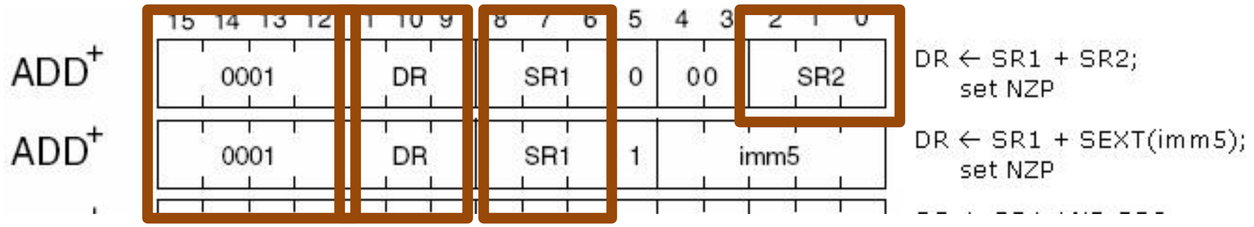
- **Fetch Phase Operations:**
  - MAR ← PC, PC ← PC + 1
  - MDR ← M[MAR] (several clock cycles)
  - IR ← MDR



Instruction Cycle: Decode Phase

- **Decode Phase:** decode word in IR as instruction
  - **Evaluate address**
  - **Fetch operands**

  - Use opcode to determine next state of control unit
    - Example: ADD operation – next state is 1 (PC will be PC+1)



	15 14 13 12	11 10 9	8 7 6	5	4 3	2 1 0	
ADD+	0001	DR	SR1	0	00	SR2	DR ← SR1 + SR2; set NZP
ADD+	0001	DR	SR1	1	imm5		DR ← SR1 + SEXT(imm5); set NZP

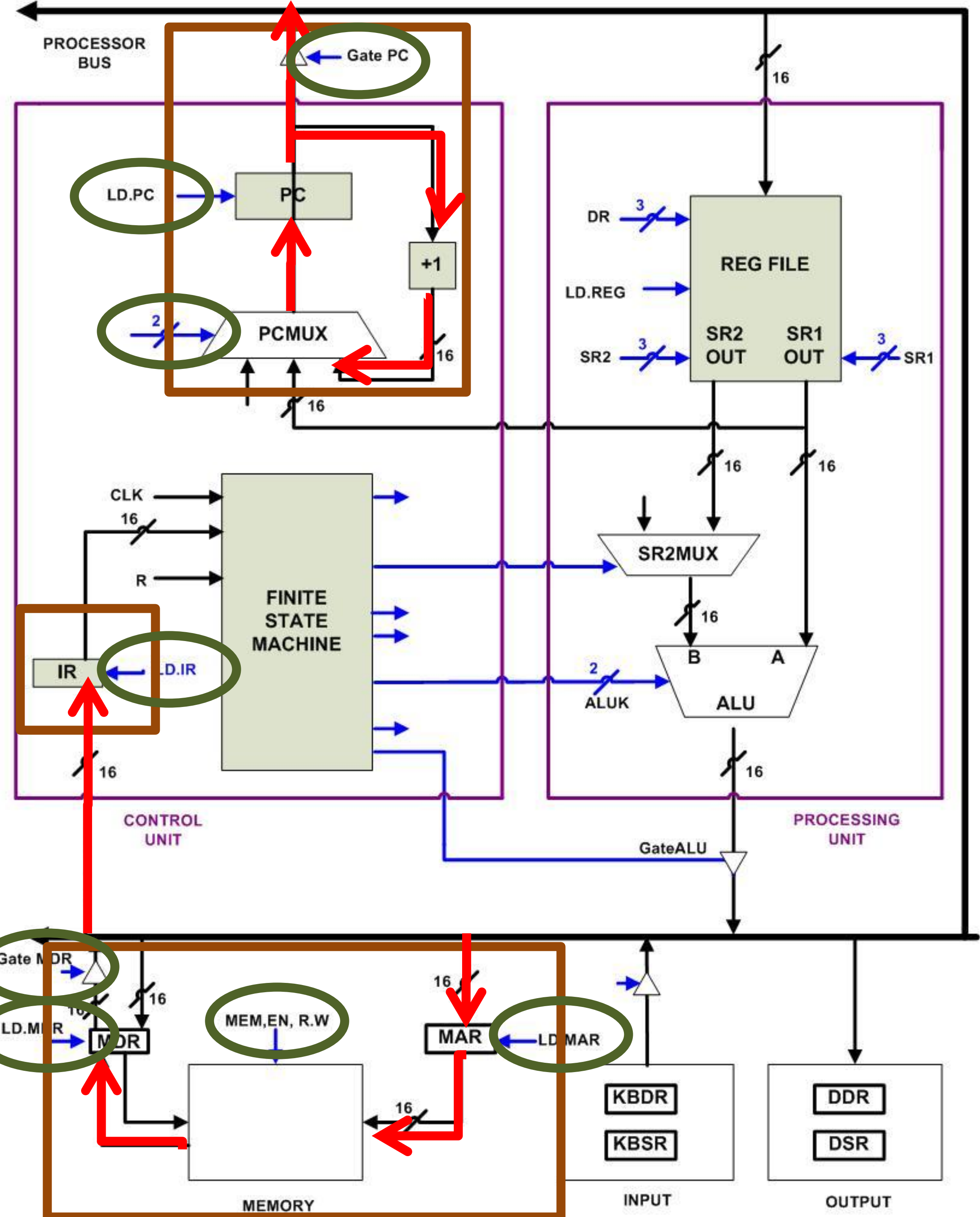
Instruction Cycle: Execute Phase

- **Execute Phase:** execute instruction
  - Move data across data path to function unit
  - Activate function unit (includes Arithmetic/Logic Unit (ALU))
  - Set status bits (condition codes)
  - **Store results**

- **Example:** ADD ~ R4 $\leftarrow$ R5 + R6
  - Control signals to move R5, R6 to ALU
  - Control signals to compute addition
  - Set NZP
  - Control signals to move the result from ALU to R4

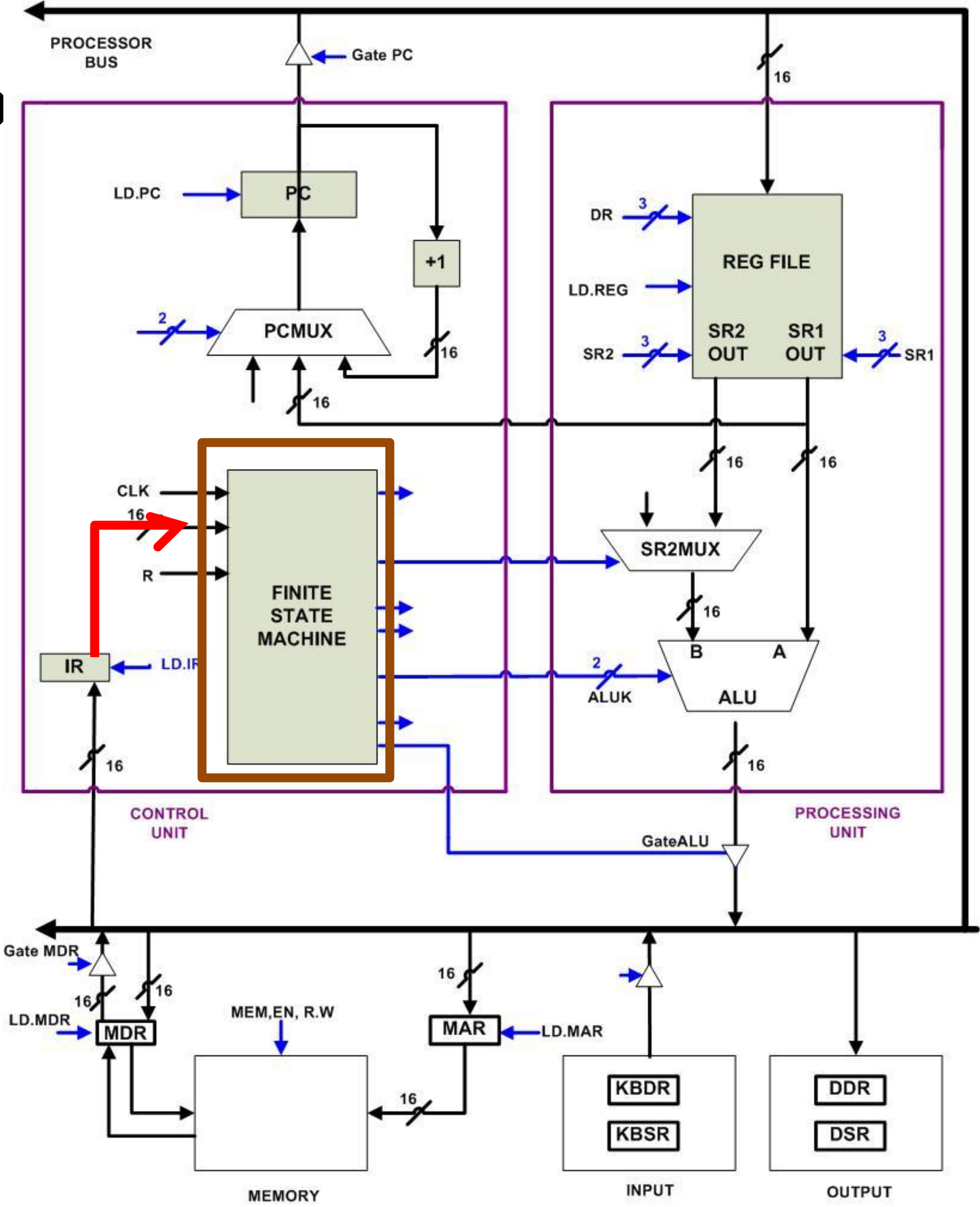
Fetch Word

- **Wanted:** Move bits and words at the right time and to/from the right locations
- Fetch phase
  - Control signals?

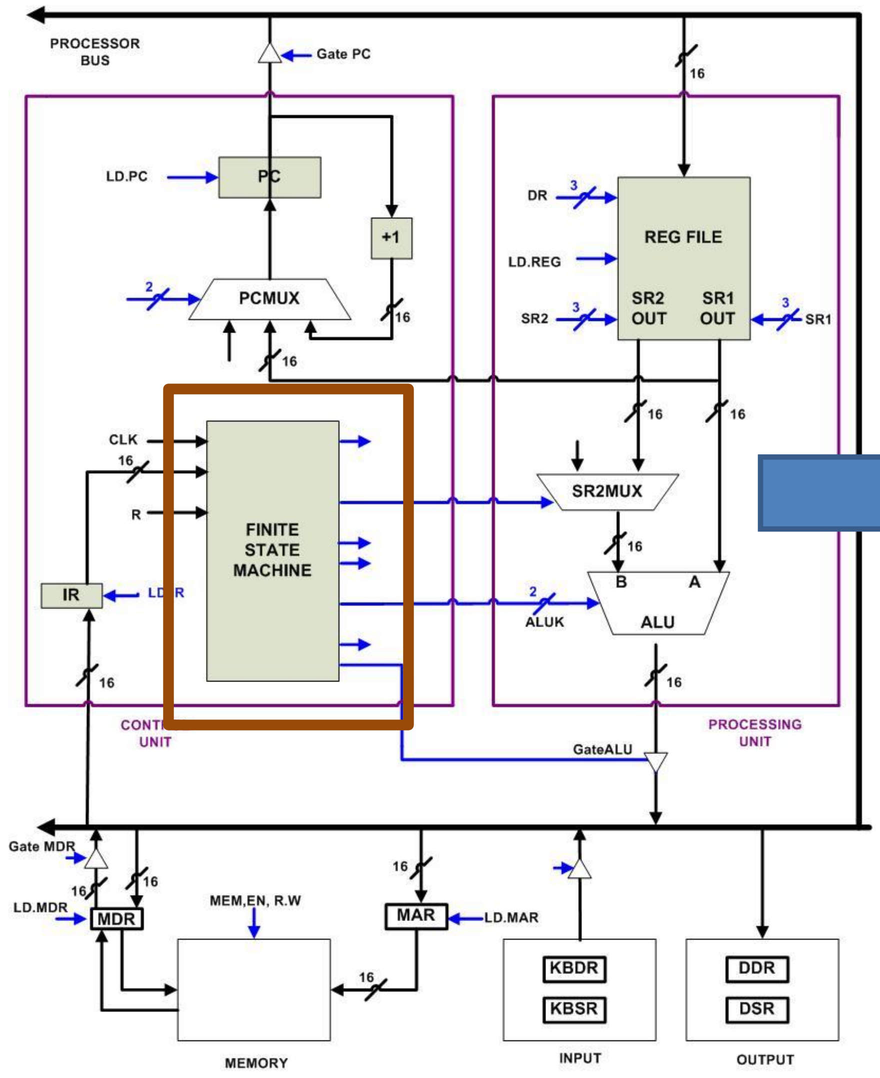


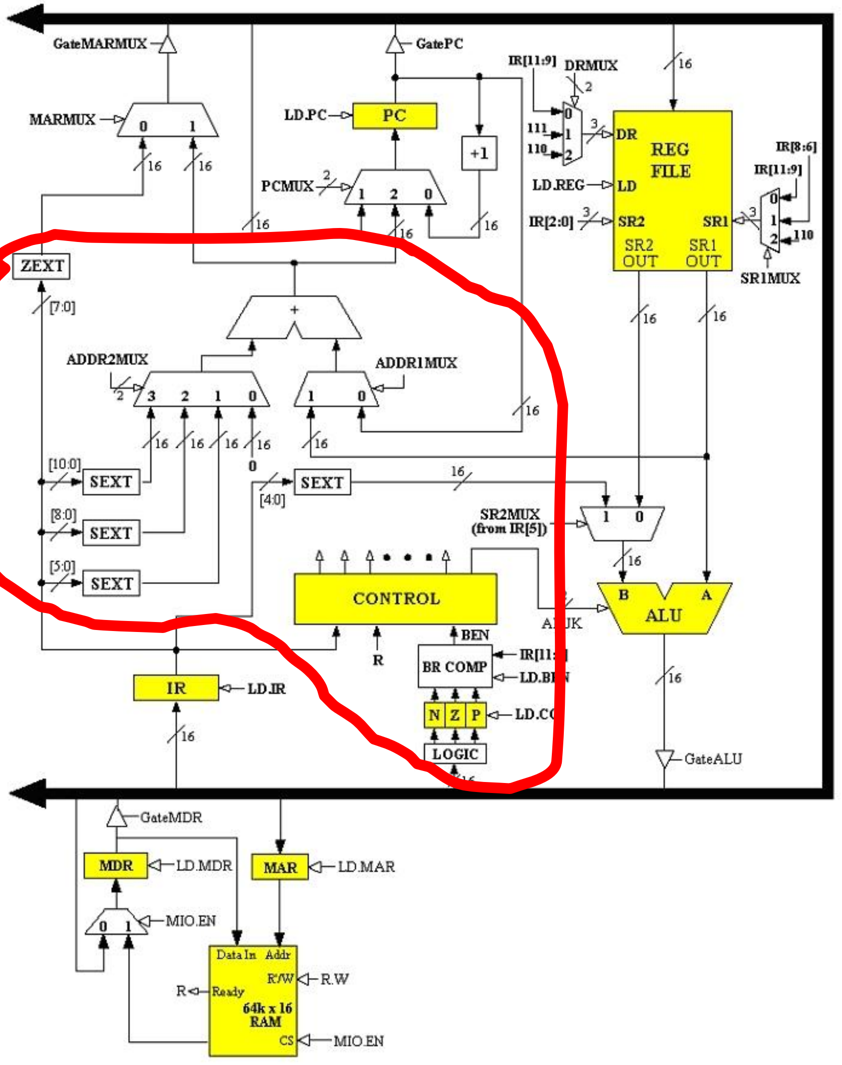
Deco

- Decode phase
  - Control signals?
- Execute phase:
  - Control signals?

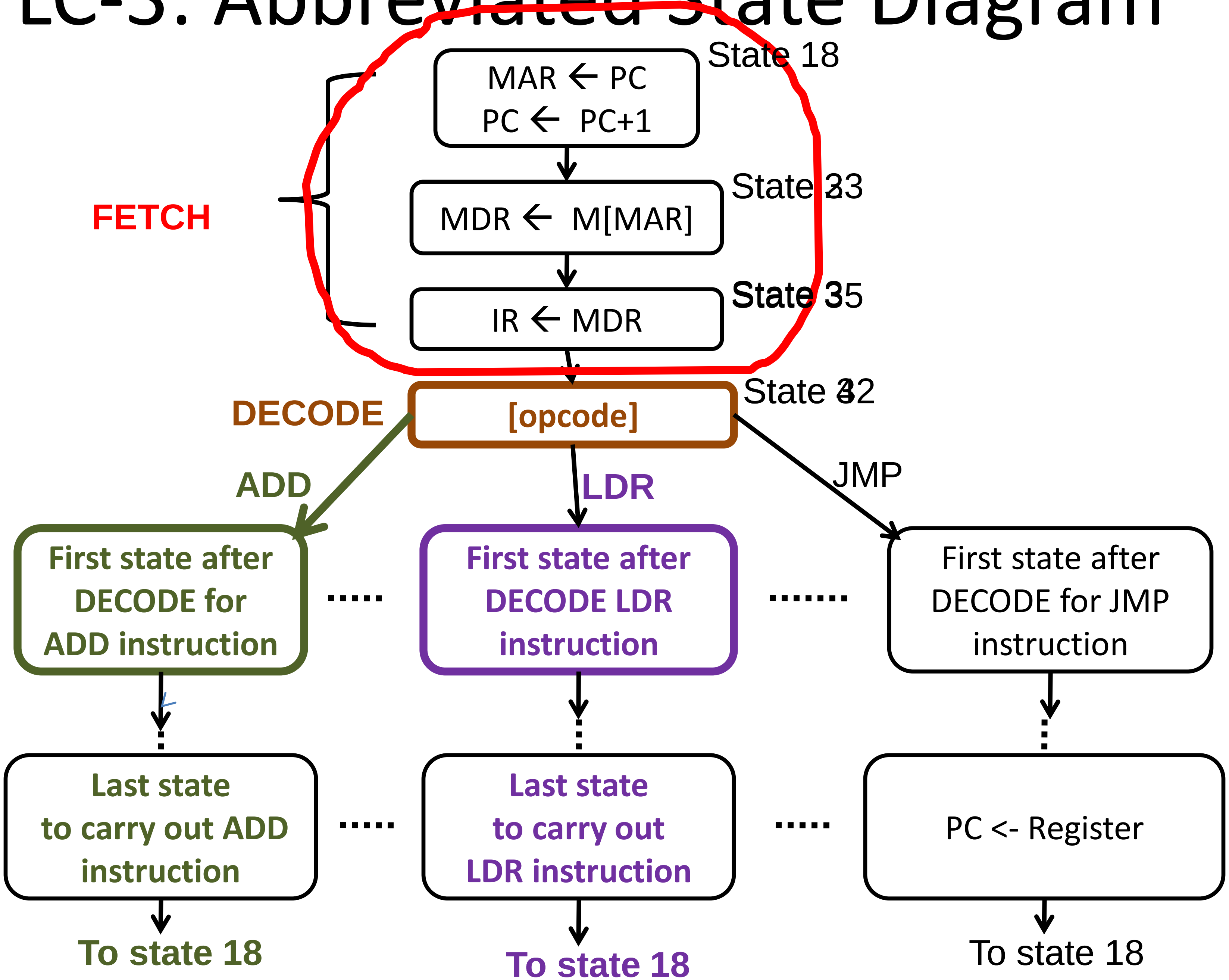


Design of Finite State Machine





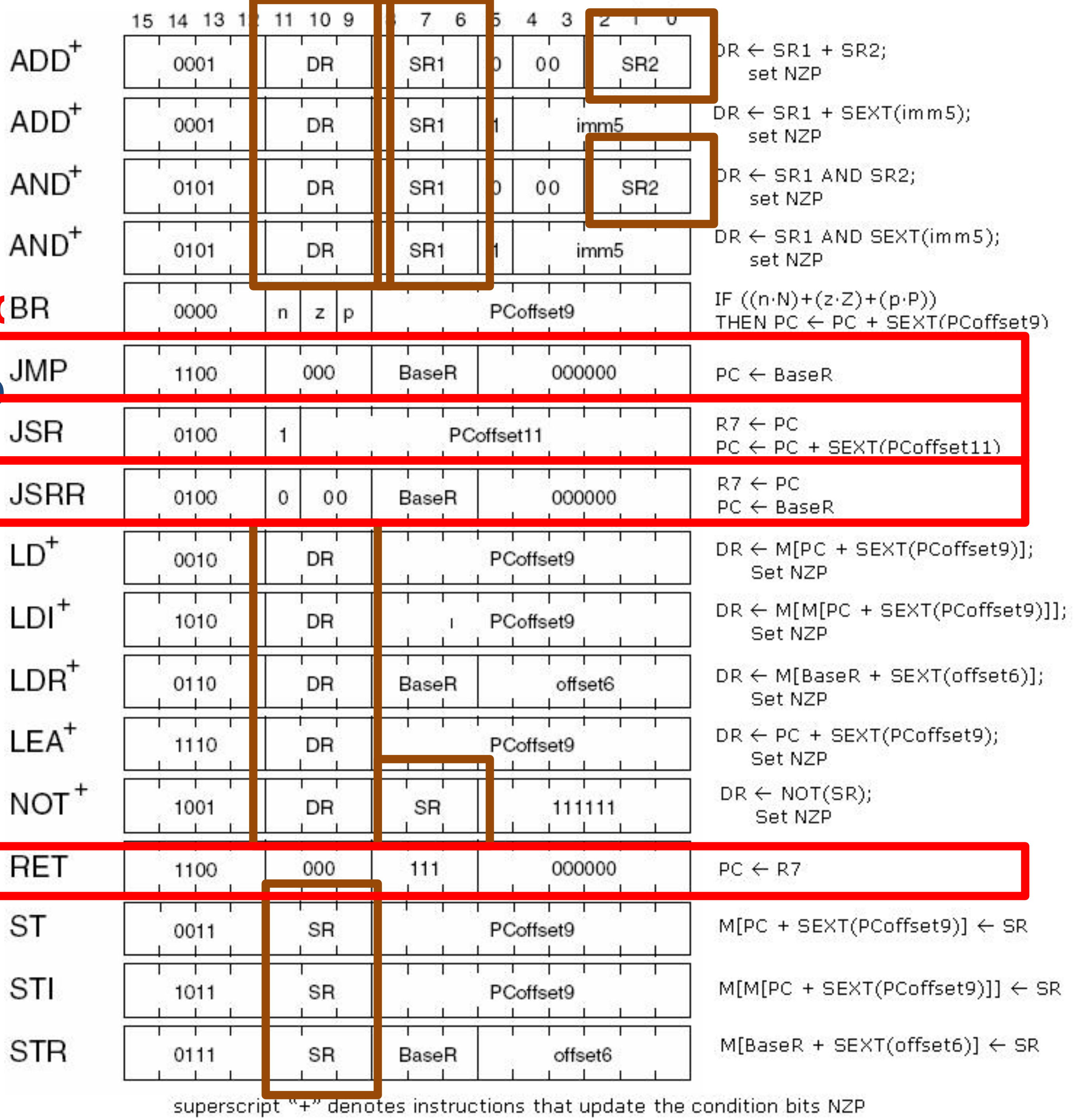
LC-3: Abbreviated State Diagram



State 18
MAR ← PC
PC ← PC+1
FETCH
State 33
MDR ← M[MAR]
State 35
IR ← MDR
State 32
DECODE
[opcode]
ADD
LDR
JMP
First state after DECODE for ADD instruction
First state after DECODE LDR instruction
First state after DECODE for JMP instruction
Last state to carry out ADD instruction
Last state to carry out LDR instruction
PC <- Register
To state 18
To state 18
To state 18

Where is the c

Wanted: deco



	15 14 13 12	11 10 9	8 7 6	5 4 3 2 1 0	
ADD$^+$	0001	DR	SR1	0 00 SR2	DR ← SR1 + SR2; set NZP
ADD$^+$	0001	DR	SR1	1 imm5	DR ← SR1 + SEXT(imm5); set NZP
AND$^+$	0101	DR	SR1	0 00 SR2	DR ← SR1 AND SR2; set NZP
AND$^+$	0101	DR	SR1	1 imm5	DR ← SR1 AND SEXT(imm5); set NZP
BR	0000	n z p	PCoffset9		IF ((n·N)+(z·Z)+(p·P)) THEN PC ← PC + SEXT(PCoffset9)
JMP	1100	000	BaseR	000000	PC ← BaseR
JSR	0100	1 PCoffset11			R7 ← PC PC ← PC + SEXT(PCoffset11)
JSRR	0100	0 00	BaseR	000000	R7 ← PC PC ← BaseR
LD$^+$	0010	DR	PCoffset9		DR ← M[PC + SEXT(PCoffset9)]; Set NZP
LDI$^+$	1010	DR	PCoffset9		DR ← M[M[PC + SEXT(PCoffset9)]]; Set NZP
LDR$^+$	0110	DR	BaseR	offset6	DR ← M[BaseR + SEXT(offset6)]; Set NZP
LEA$^+$	1110	DR	PCoffset9		DR ← PC + SEXT(PCoffset9); Set NZP
NOT$^+$	1001	DR	SR	111111	DR ← NOT(SR); Set NZP
RET	1100	000	111	000000	PC ← R7
ST	0011	SR	PCoffset9		M[PC + SEXT(PCoffset9)] ← SR
STI	1011	SR	PCoffset9		M[M[PC + SEXT(PCoffset9)]] ← SR
STR	0111	SR	BaseR	offset6	M[BaseR + SEXT(offset6)] ← SR

superscript "+" denotes instructions that update the condition bits NZP

Decoding Operands

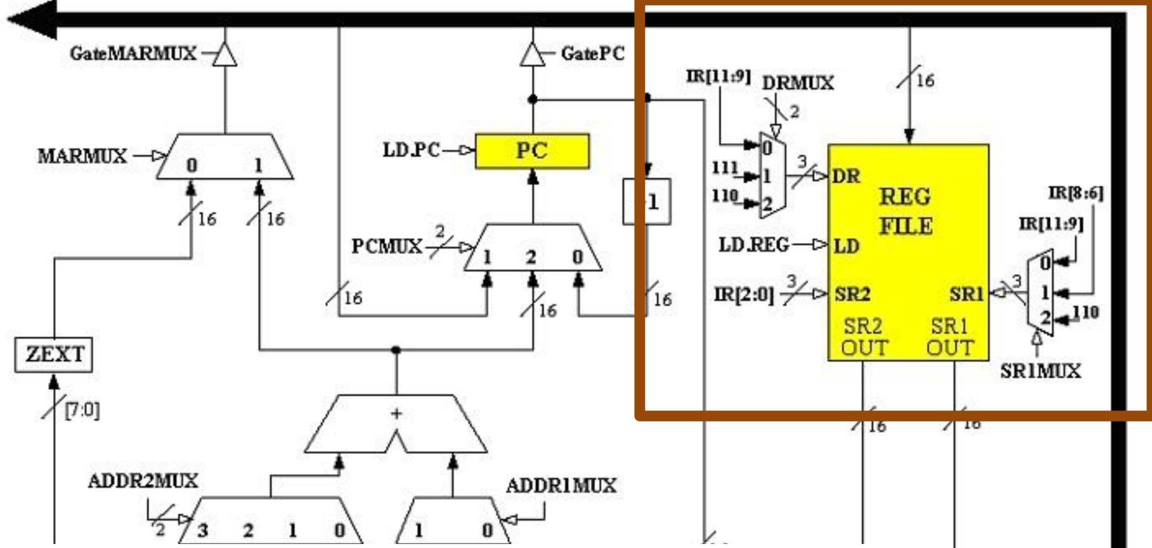
Where is the operand?

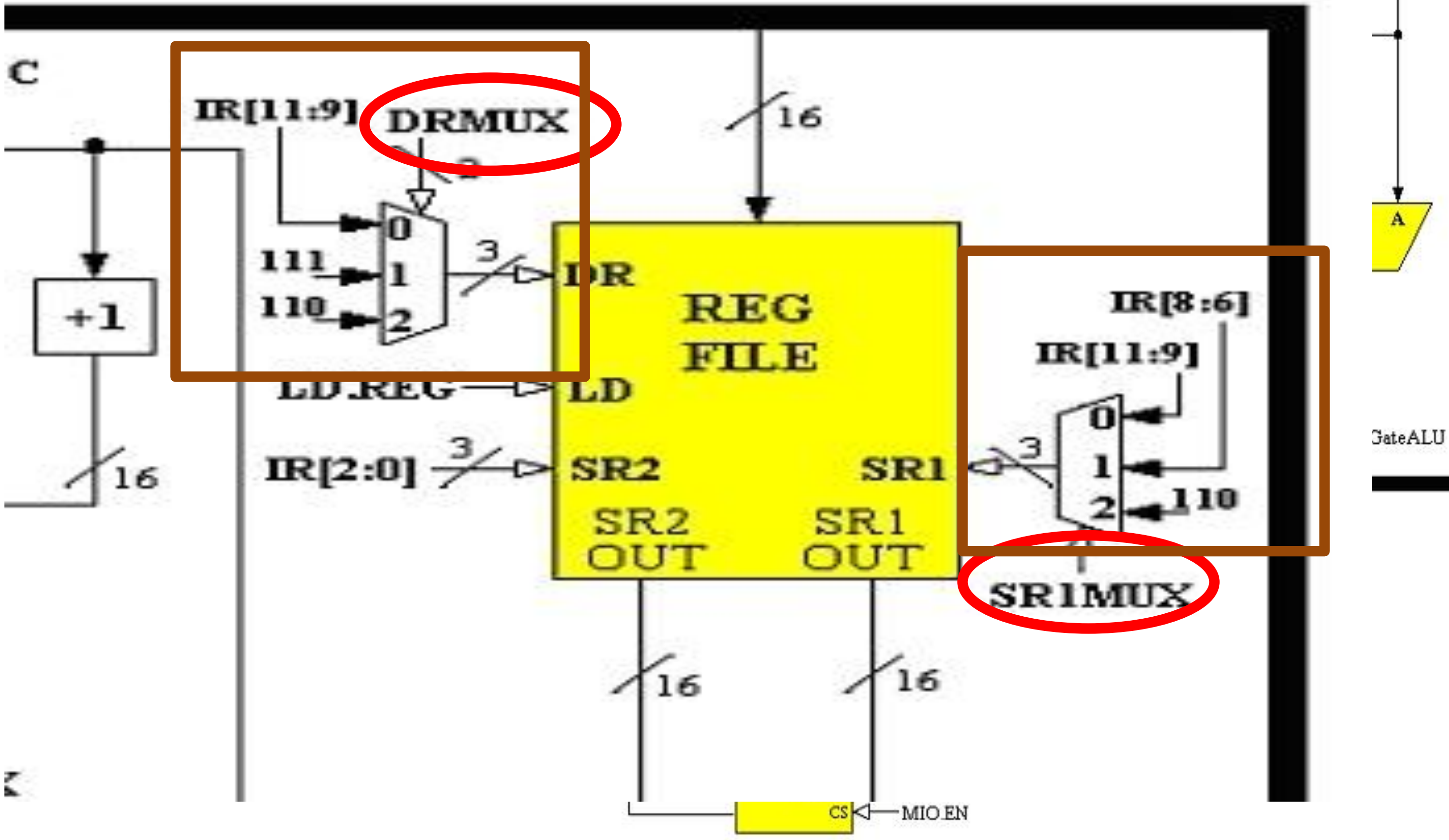
Wanted: decode operand addresses and values

- Direct Decode: SR2 input: IR[2:0]
- **Need MUX:**
  - **SR1 input: IR[11:9] or IR[8:6] or 110**
  - **DR input: IR[11:9] or 111 or 110**
- Note:
  - R6 (110) ~ stack pointer
  - R7 (111) ~ used for subroutine calls

Control Sign

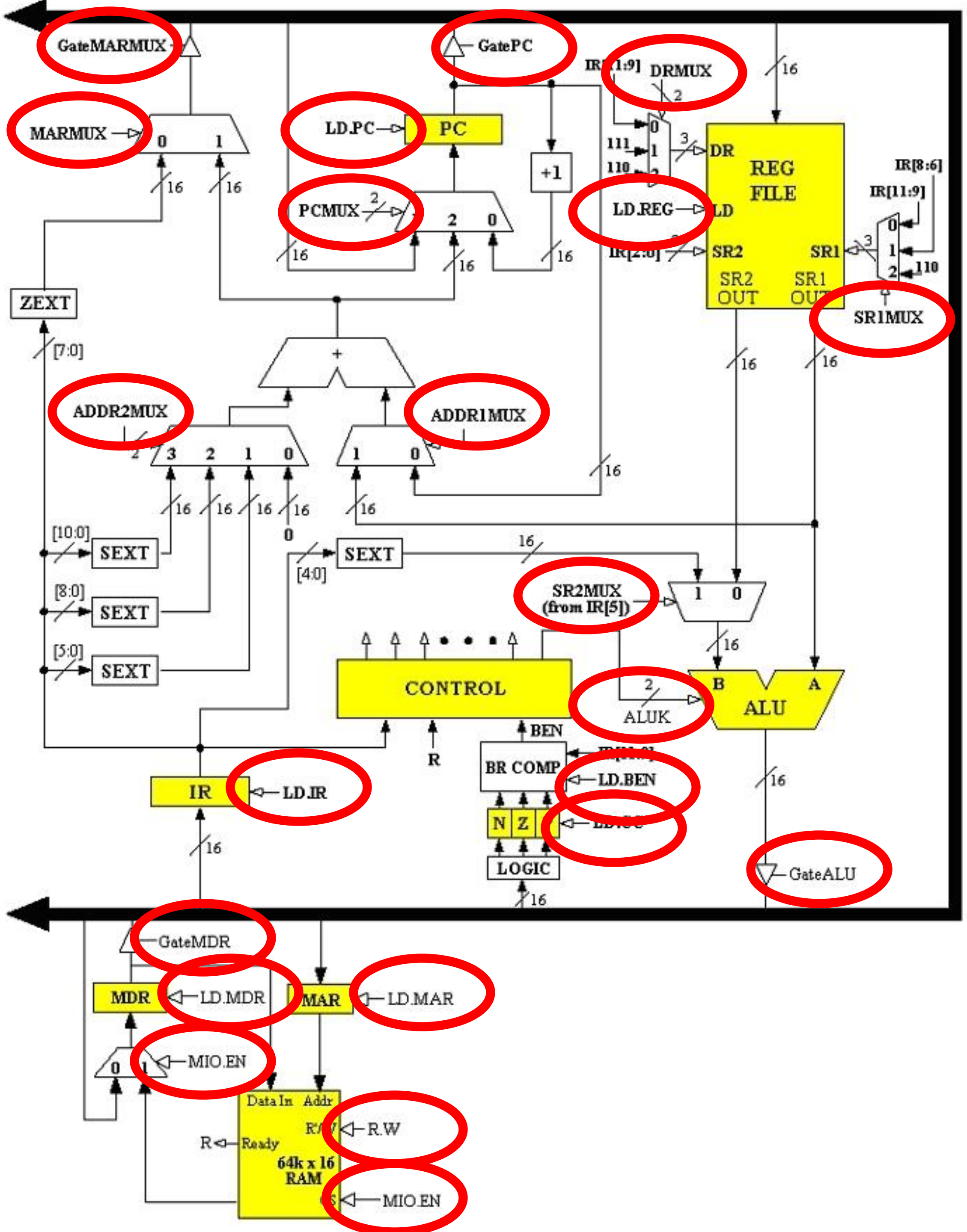
- Where are the inpu
  architecture?





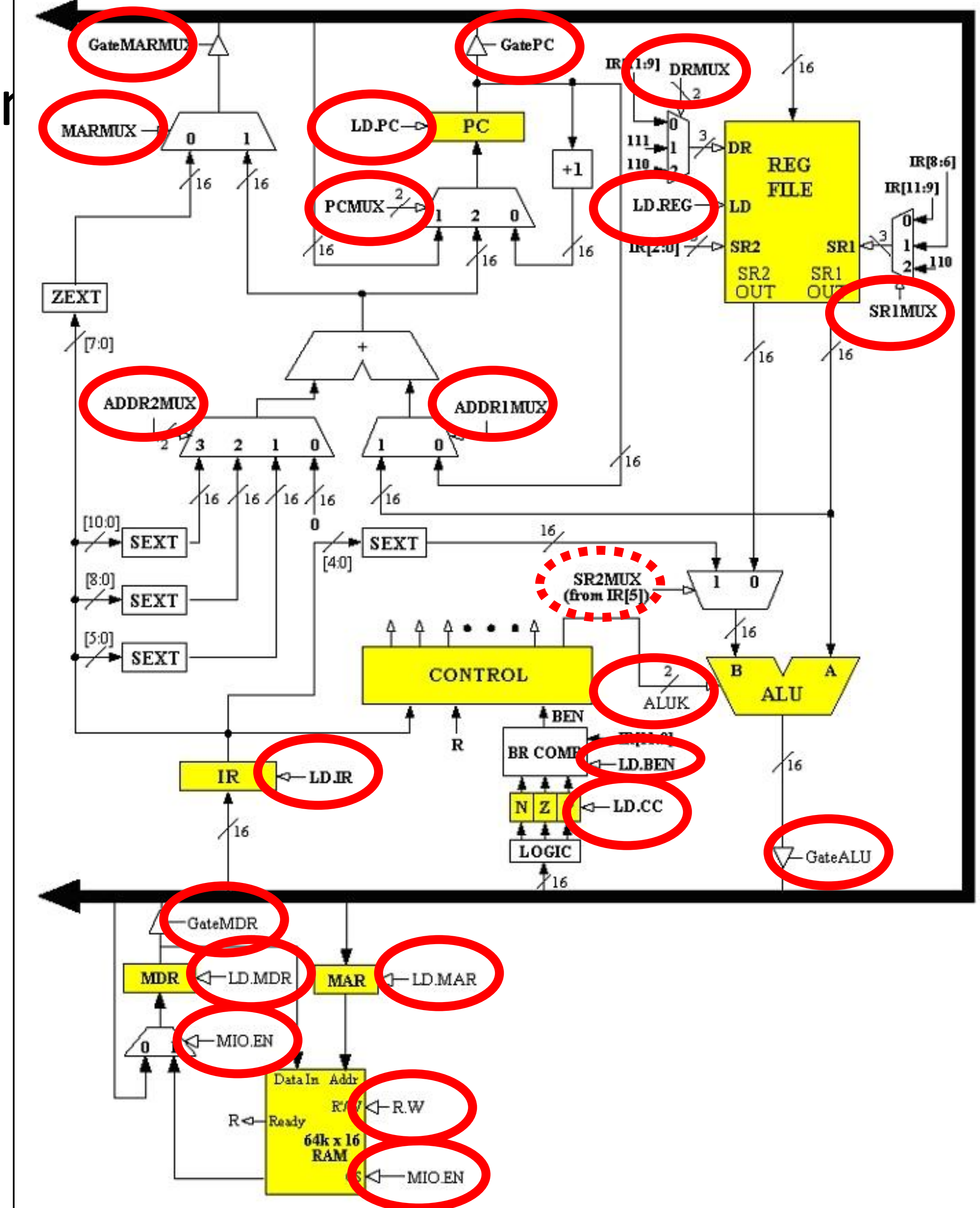
Control Sign

- Where are all **inpu** architecture?



Control Signals i

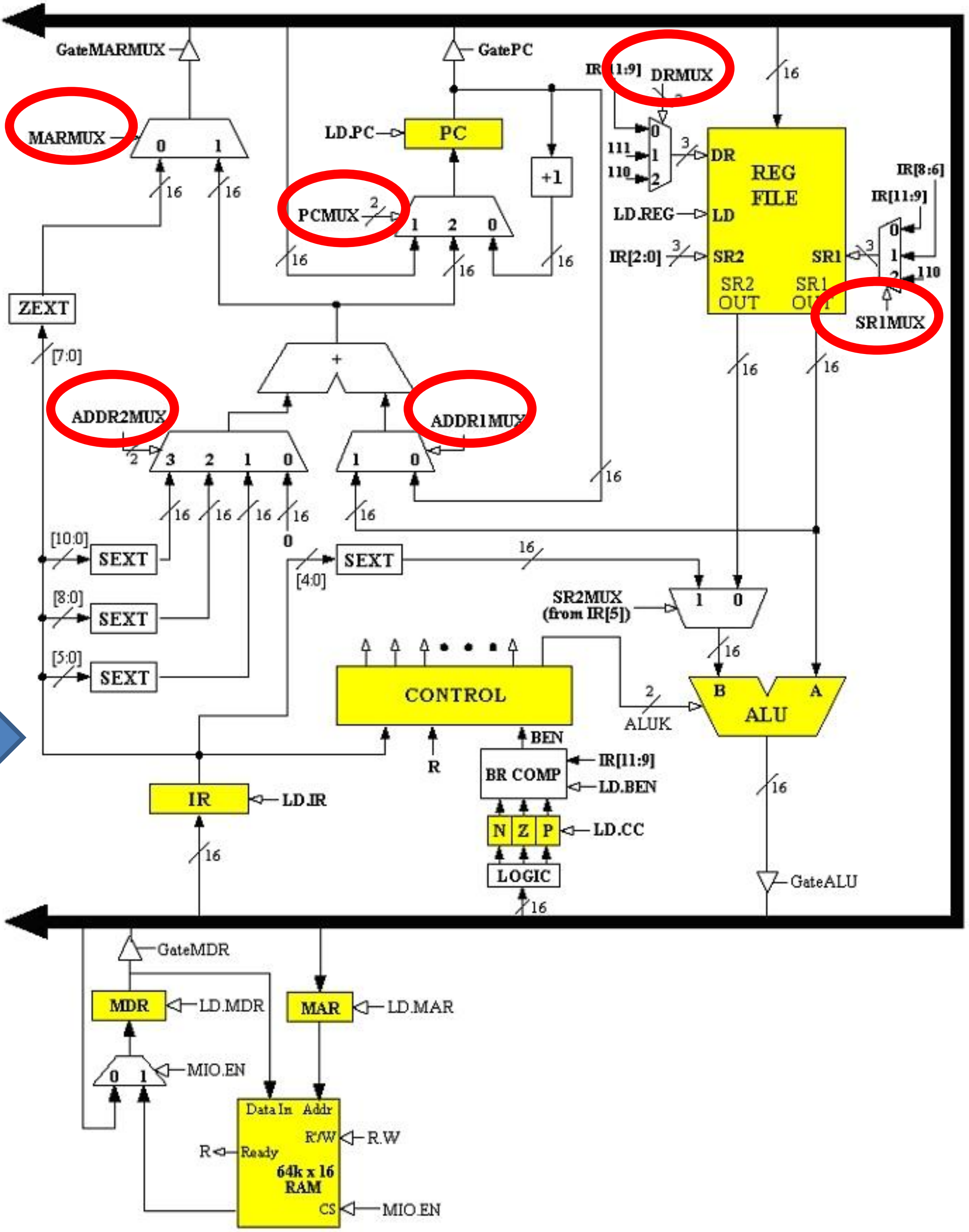
- Summary:
  - GateXX: 4
  - LD.XX: 7
  - xxMUX: 6 (+1 IR[5])
  - ALUK: 1
  - MIO.EN: 1
  - R.W: 1



Control Signals ir

Control Signal	Number of Signals	Total Number of Bits for Control Signals
GateXX	4	4
LD.XX	7	7
xxMUX	6	10
ALUK	1	2
MIO.EN	1	1
R.W	1	1

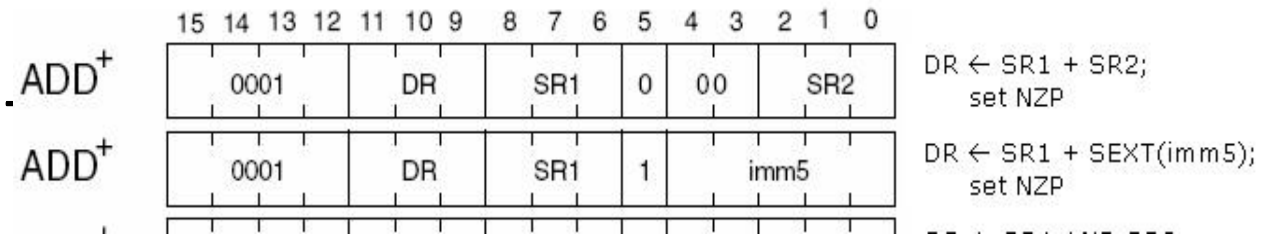
Generate: 25 bits of control signals!!!



Execute Phase: ADD Operation

- Task: execute ADD (1 microinstruction)
  - SR1 (DR) ← SR1 + SR2, Load NZP
  - SR1 (DR) ← SR1 + #2, Load NZP

-



	15 14 13 12	11 10 9	8 7 6	5	4 3	2 1 0	
ADD$^+$	0001	DR	SR1	0	00	SR2	DR ← SR1 + SR2; set NZP
ADD$^+$	0001	DR	SR1	1	imm5		DR ← SEXT(imm5); set NZP

0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0	1	1	0	0	0	1	0

What are the control signals needed for the execution of the microinstruction?

Execute Phase: ADD Operation – Bus Gate Signals

- Task: SR1 (DR) ← SR1 + SR2, Load NZP

-

0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0

  - GateXX:
    - GatePC=0
    - GateMARMUX = 0
    - GateMDR = 0
    - GateALU = 1

Control Signal	Number of Signals	Total Number of Bits for Control Signals
GateXX	4	4
LD.XX	7	7
xxMUX	6	10
ALUK	1	2
MIO.EN	1	1
RW	1	1

Execute Phase: ADD Operation – Load signals

- Task: SR1 (DR) ← SR1 + SR2, Load NZP

-

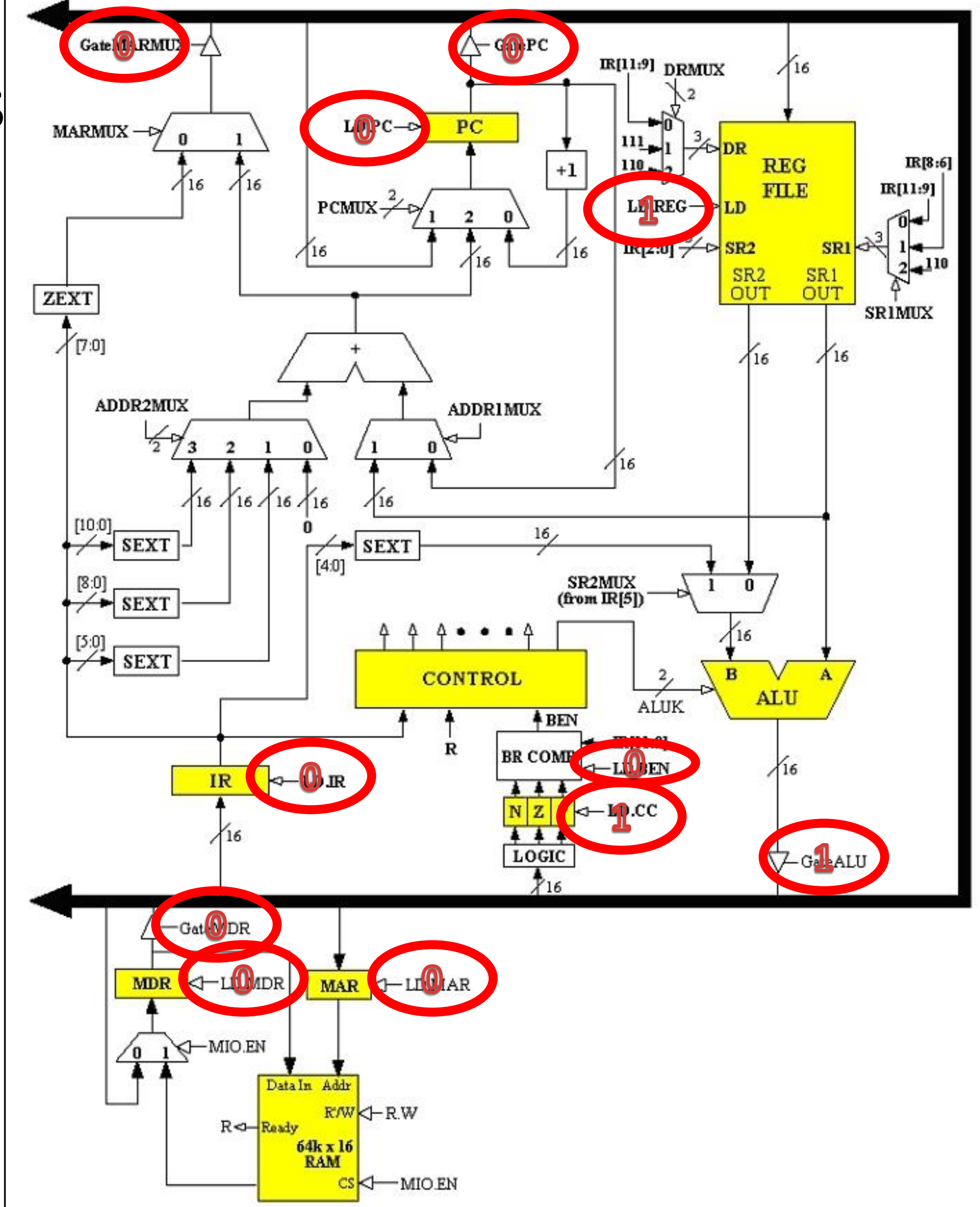
0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0

  – LD.XX:
    - LD.PC=0
    - LD.BEN = 0
    - LD.MAR = 0
    - LD.MDR = 0
    - LD.IR=0
    - LD.REG =1
    - LD.CC = 1

Control Signal	Number of Signals	Total Number of Bits for Control Signals
GateXX	4	4
LD.XX	7	7
xxMUX	6	10
ALUK	1	2
MIO.EN	1	1
RW	1	1

Control Signals

- Current Assignment of Control Signals:
  - GateXX:
  - LD.XX:



Execute Phase: ADD Operation – Multiplexer Selection Signals

- Task: SR1 (DR) ← SR1 + SR2, Load NZP

0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0

SR2MUX=IR[5] = 0

- Control signals:
  - xxMUX:
    - SR1MUX = 01 (IR[8:6])
    - DRMUX = 00 (IR[11:9])
    - MARMUX =x
    - PCMUX =x
    - ADDR1MUX =x
    - ADDR2MUX=x

Control Signal	Number of Signals	Total Number of Bits for Control Signals
GateXX	4	4
LD.XX	7	7
xxMUX	6	10
ALUK	1	2
MIO.EN	1	1
RW	1	1

Execute Phase: ADD Operation – Other Control Signals

- Task: SR1 (DR) ← SR1 + SR2, Load NZP

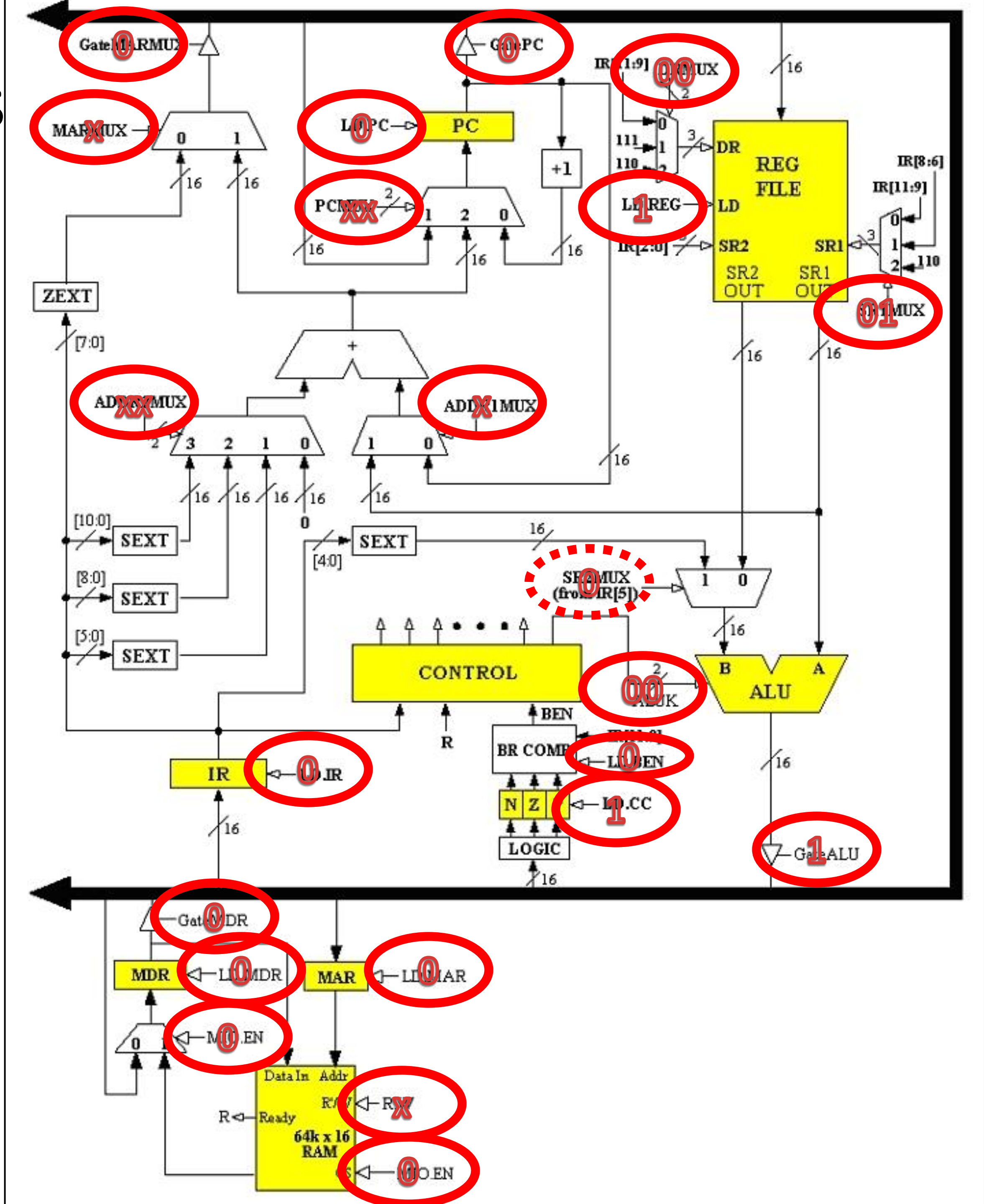
0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0

- Control signals:
  - ALUK = 00
  - MIO.EN =0
  - RW =x

Control Signal	Number of Signals	Total Number of Bits for Control Signals
GateXX	4	4
LD.XX	7	7
xxMUX	6	10
ALUK	1	2
MIO.EN	1	1
RW	1	1

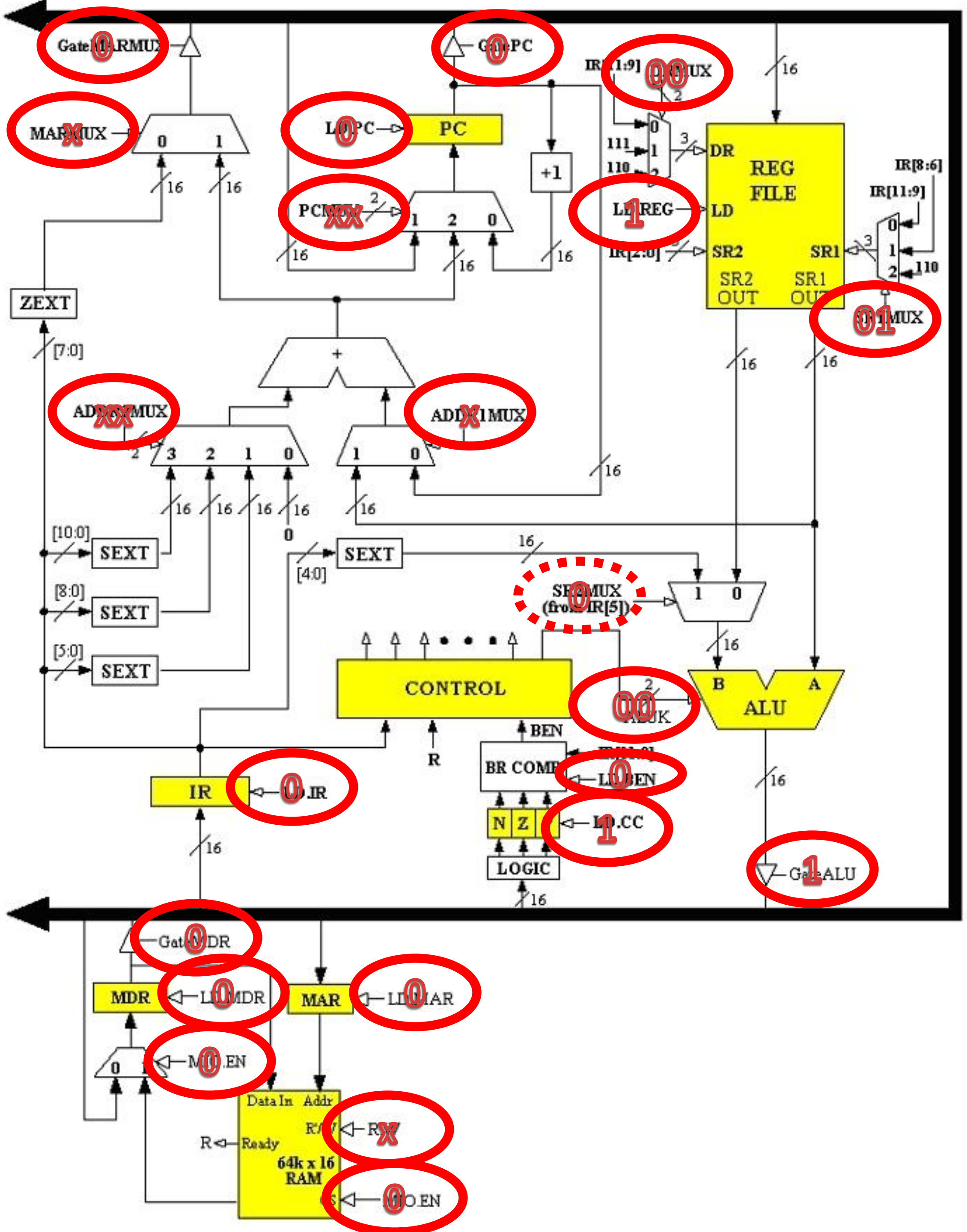
Control Signals

- All Assignments of Control Signals:
  - GateXX:
  - LD.XX:
  - xxMUX:
  - ALUK:
  - MIO.EN:
  - R.W:



Control Signals

- Selections based on control signals:
  - SR1 selected based on IR[8:6]
  - SR2 selected based on IR[2:0]
  - DR selected based on IR[11:9]
  - What to add selected based on IR[5]



Execute Phase: ADD Operation – Summary of Control Signals

Signal Name	Binary Value
GateMARMUX	0
GateMDR	0
GateALU	1
GatePC	0

Signal Name	Binary Value
ALUK	00
MIO.EN	0
R.W	X

Signal Name	Binary Value
LD.BEN	0
LD.MAR	0
LD.MDR	0
LD.IR	0
LD.PC	0
LD.REG	1
LD.CC	1

Signal Name	Binary Value
MARMUX	X
PCMUX	XX
ADDR1MUX	X
ADDR2MUX	XX
DRMUX	00
SR1MUX	01

LC-3 Control Word Fields

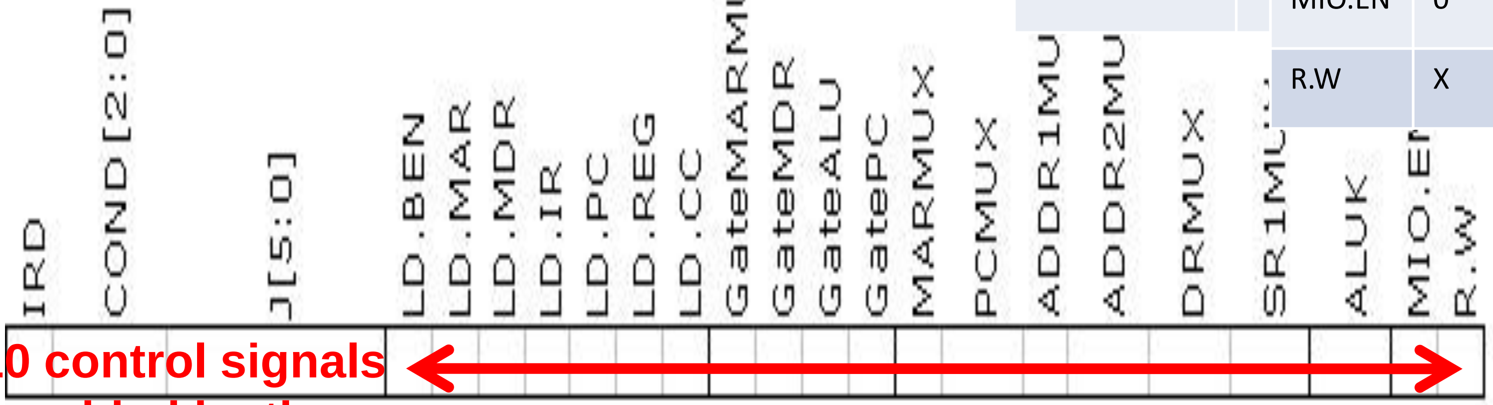
- **no Interrupt or Exception Control signals!**

Signal Name	Binary Value
LD.BEN	0
LD.MAR	0
LD.MDR	0
LD.IR	0
LD.PC	0
LD.REG	1
LD.CC	1

Signal Name	Binary Value
GateMARMUX	0
GateMDR	0
GateALU	1
GatePC	0

Signal Name	Binary Value
MARMUX	X
PCMUX	XX
ADDR1MUX	X
ADDR2MUX	X
DRMUX	0
SR1MUX	0

Signal Name	Binary Value
ALUK	00
MIO.EN	0
R.W	X



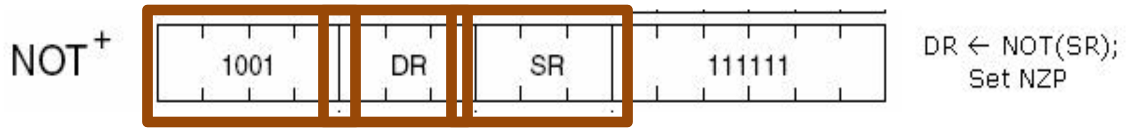
For a given state

10 control signals provided by the current clock cycle

25 control signals

Execute Phase: NOT Operation

- Task: SR1 (DR) ← NOT (SR1), Load NZP



1	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1

- Directly derived control signals:
  - SR2MUX=1 (IR[5])
  - SR1MUX = 01 (IR[8:6])
  - DRMUX = 00 (IR[11:9])

Execute Phase: NOT Operation – Summary of Control Signals

- Control signals

Signal Name	Binary Value
GateMARMUX	0
GateMDR	0
GateALU	1
GatePC	0

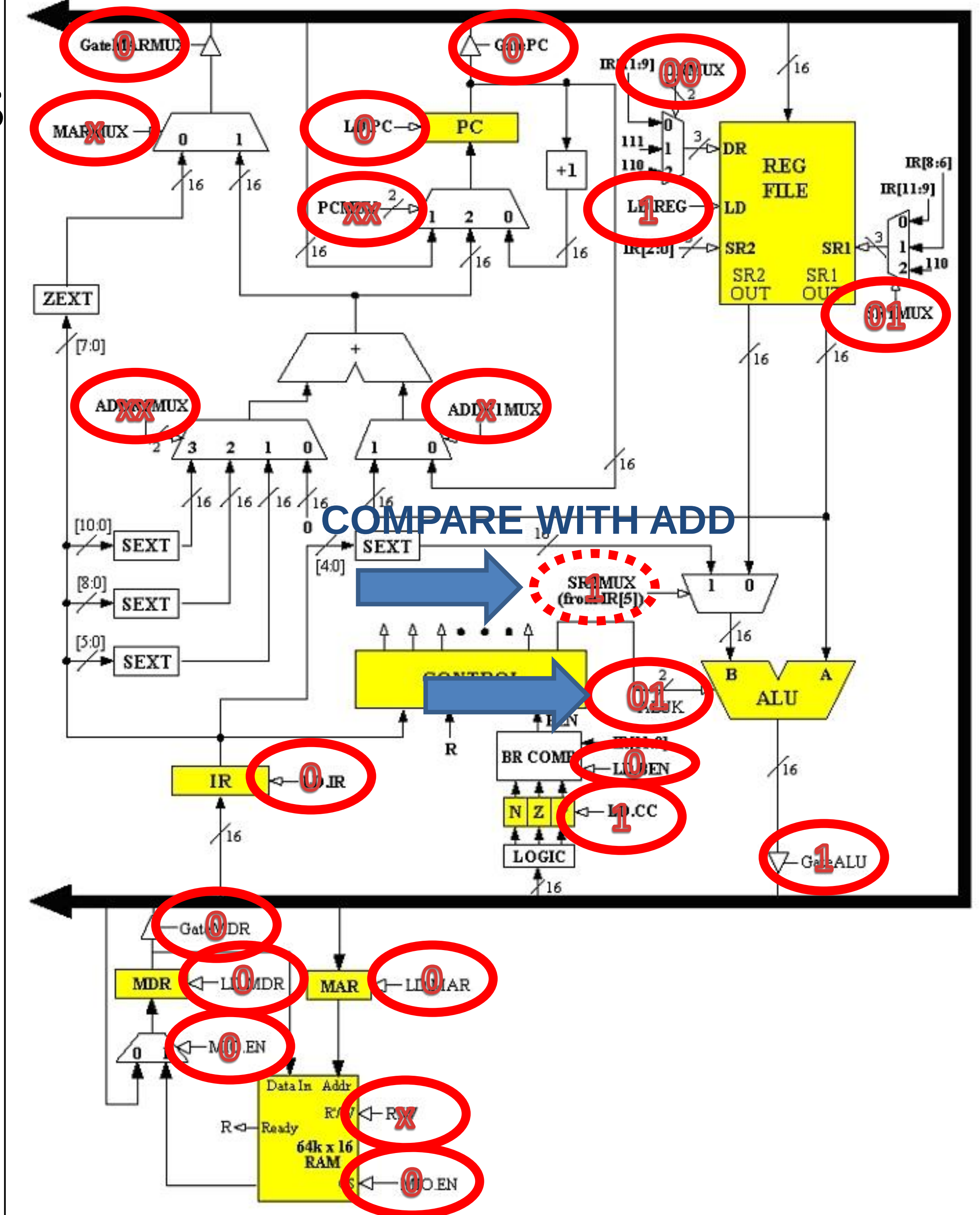
Signal Name	Binary Value
ALUK	10
MIO.EN	0
R.W	x

Signal Name	Binary Value
LD.BEN	0
LD.MAR	0
LD.MDR	0
LD.IR	0
LD.PC	0
LD.REG	1
LD.CC	1

Signal Name	Binary Value
MARMUX	X
PCMUX	XX
ADDR1MUX	X
ADDR2MUX	XX
DRMUX	00
SR1MUX	01

Control Signals

- Selections based on control signals:
  - SR1 selected based on IR[8:6]
  - DR selected based on IR[11:9]
  - What to add selected based on IR[5]



GateMARMUX 0
MARMUX x
GatePC 0
LD.PC 0
PC
PCMUX xx
DRMUX 00
LD.REG 1
REG FILE
SR1MUX 01
ADDR2MUX xx
ADDR1MUX x
COMPARE WITH ADD
SR2MUX (from IR[5]) 1
ALUK 01
CONTROL
LD.IR 0
IR
LD.BEN 0
BR COMP
LD.CC 1
LOGIC
GateALU 1
ALU
GateMDR 0
LD.MDR 0
MDR
LD.MAR 0
MAR
MIO.EN 0
R.W x
MIO.EN 0
64k x 16 RAM
ZEXT
SEXT