



DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE



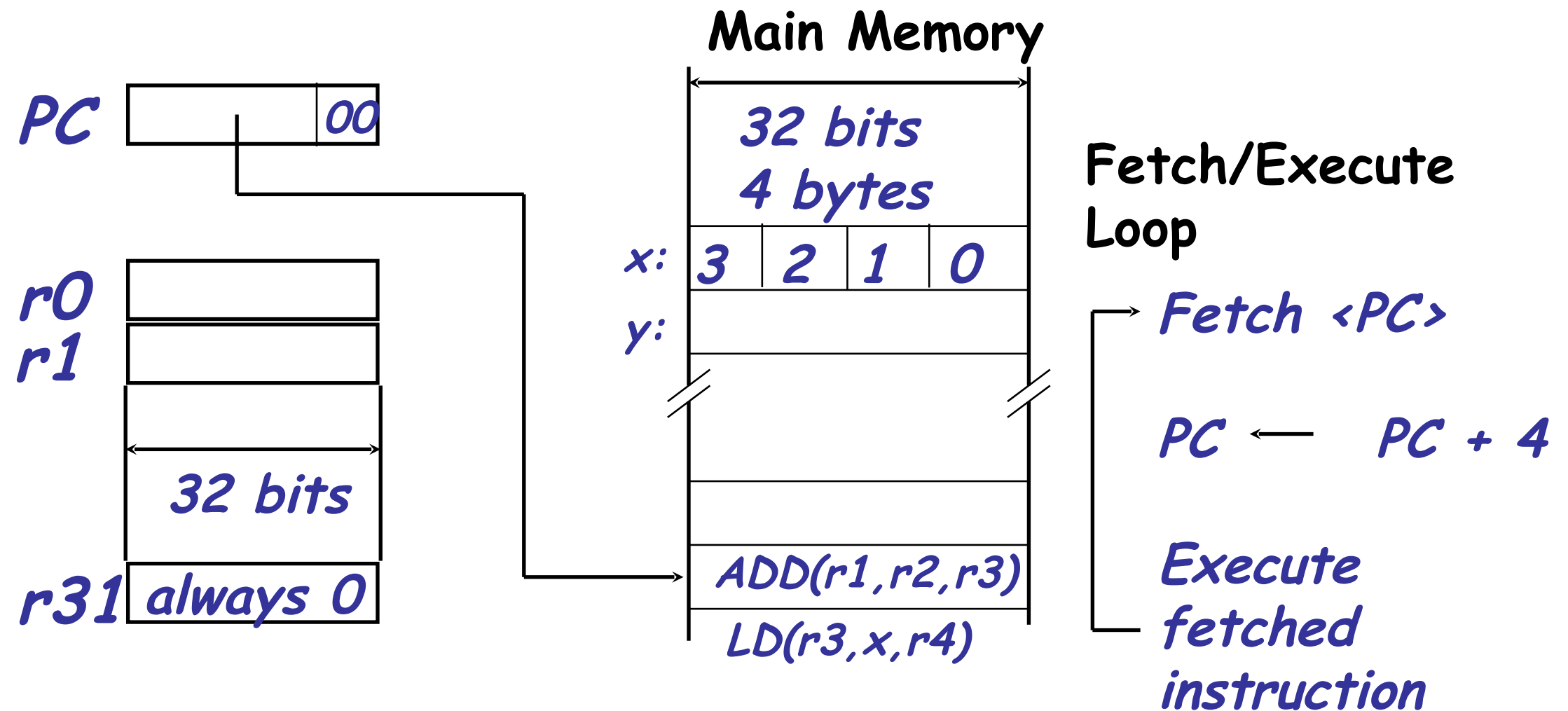
```
0010111010100011101011110010011101010101001000101
11010101110101010000101010101001010101010101010
10100101001001001010101010101010101010101010101010
11100001111010110000000111101010101010000010101
1110101011100101000100101111010100010100100111010
10101001010010010010000101010110101010101010010111
0010101001010100101010000001010101001111101000011001
1000110010000111100110101011000100110101010000101010
1100101010101000010011001010100010010101010101010
10100101001001001010101010101010101010101010101010
11100001111010110000000111101010101010000010101
0010010101001010010010100100010101010101001010010
10010100100001010100100101010010100101001010010010
1001010010101001010010101001010010101001001001001
100101010101001010101010100101010101010010101010
```

										01
										02
										03
										04
										05
A	B	C	D	E	F	G	H			

Implementing the Beta in Hardware

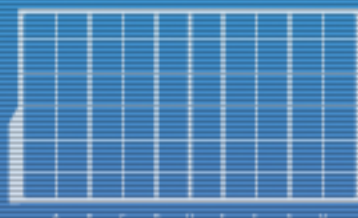
A Convenient Guide for the Final
Lab Exercise

Recap: β Visible State & Computation Model



How do we implement this?

00101010010101000011110100001100
10001100100001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
10010100101010010100101010010101

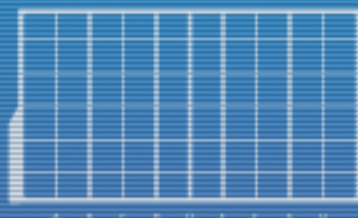


DISCS

Approach: Incremental Design

- ▶ Divide instructions into various instruction classes based on **OPCODE** field:
 - ▶ Operate instructions: **ADD, ADDC**, etc.
 - ▶ Load and Store instructions: **LD, ST**
 - ▶ Jump and Branch instructions: **JMP, BEQ**, etc.
 - ▶ Exceptions: **LDR**
- ▶ Implement datapaths for each class individually.
- ▶ Merge datapaths.

00101010010101000011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
1001010010010010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
10010101010101010101010101010101



DISCS

β Instruction Set Architecture

Form 1:

OPCODE	<i>rc</i>	<i>ra</i>	<i>rb</i>	unused
6	5	5	5	11

OPCODE = 10xxxx

ADD: $rc \leftarrow \langle ra \rangle + \langle rb \rangle$

Form 2:

OPCODE	<i>rc</i>	<i>ra</i>	literal C (signed)
6	5	5	16

OPCODE = 11xxxx

ADDC: $rc \leftarrow \langle ra \rangle + C$

OPCODE = 01xxxx

LD: $rc \leftarrow \text{MEM}[\langle ra \rangle + C] * \text{LD}(ra, C, rc)$

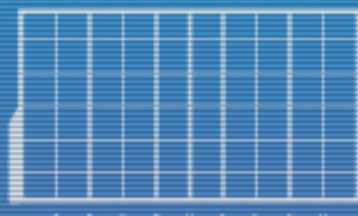
ST: $\text{MEM}[\langle ra \rangle + C] \leftarrow rc * \text{ST}(rc, C, ra)$

JMP: $rc \leftarrow \langle PC \rangle + 4; PC \leftarrow \langle ra \rangle$

BEQ: $rc \leftarrow \langle PC \rangle + 4; \text{if } \langle ra \rangle = 0 \text{ then}$
 $PC \leftarrow (\langle PC \rangle + 4) + C * 4$

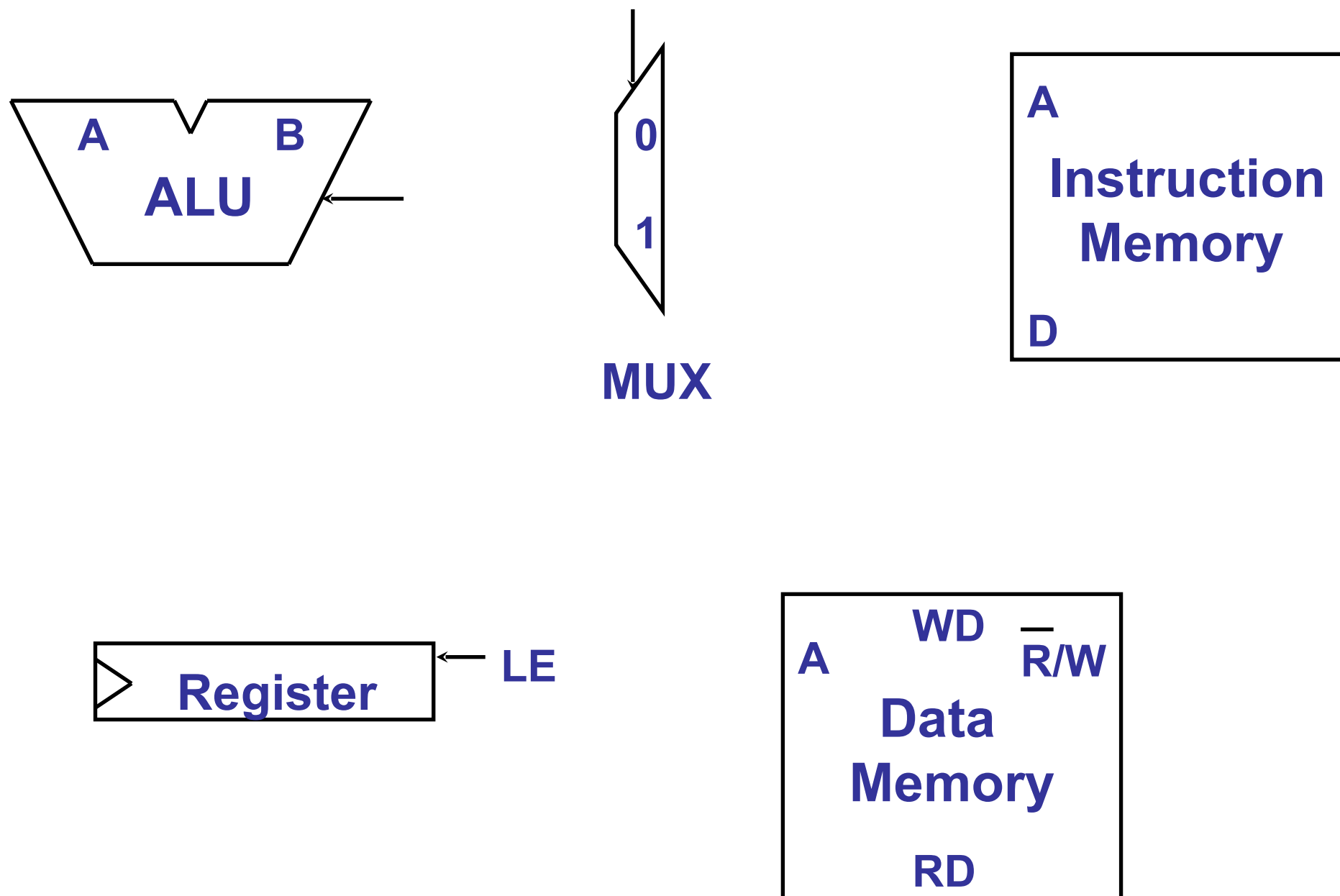
LDR: $rc \leftarrow \text{MEM}[(\langle PC \rangle + 4) + C * 4]$

00101010010101000011110100001100
 10001100100001111001101010010101
 110010101010100001001100101010100
 100101001001001010101010101010101
 11100001111010110000000111101001
 001001010100101001001010010010110
 10010100100001010100100101001010
 10010100101010010100101010010101
 10010101010101010101010101010101

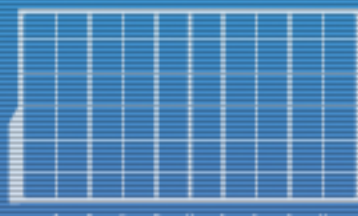


DISCS

Datapath Components



0010101001010100011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
1001010010010010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
100101010101010010101010010101

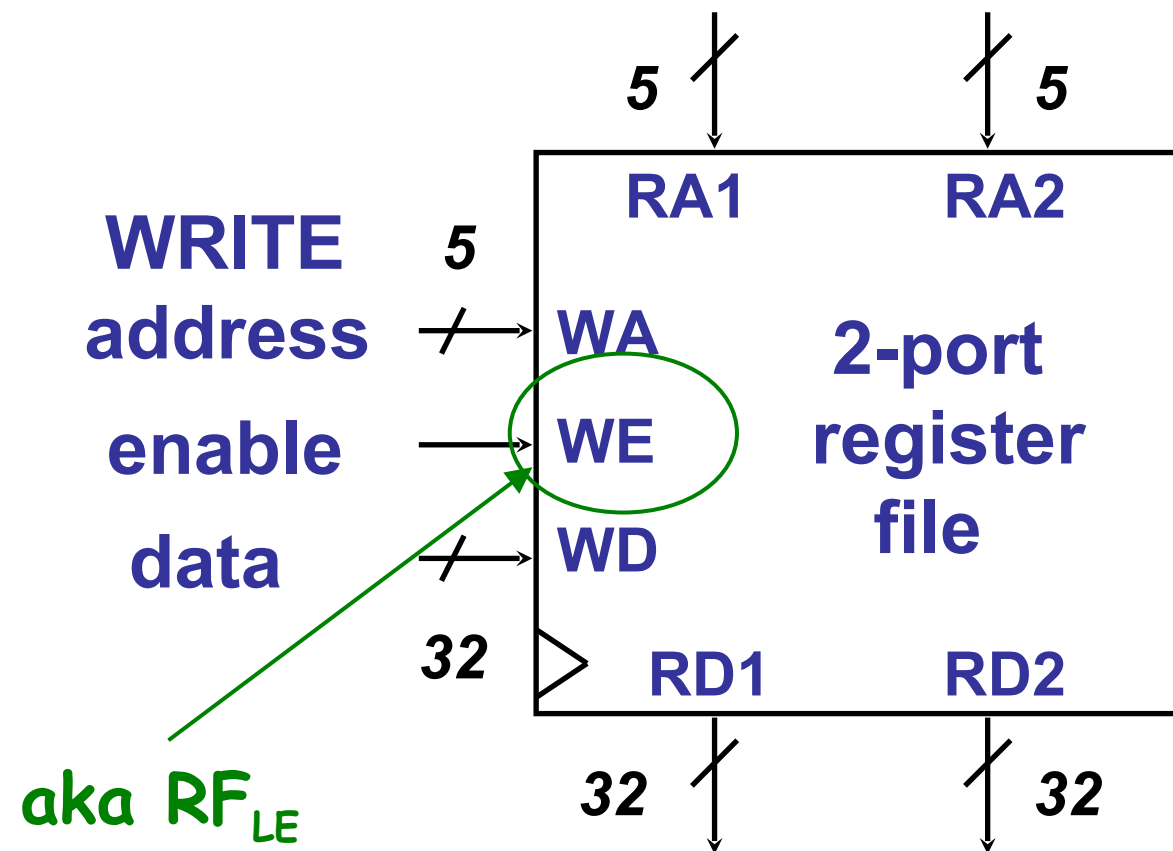


2-Port Register File

2 Combinational READ ports, 1 clocked WRITE port

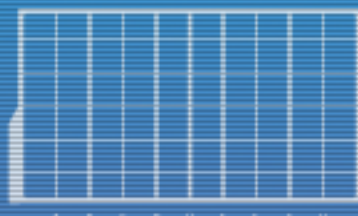
- One data input (WD), two data outputs (RD1, RD2)
- There are 3 separate ADR signals now: RA1, RA2 and WA

Independent READ addresses



Independent READ data

00101010010101000011110100001100
10001100100001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
10010101010101010101010101010101

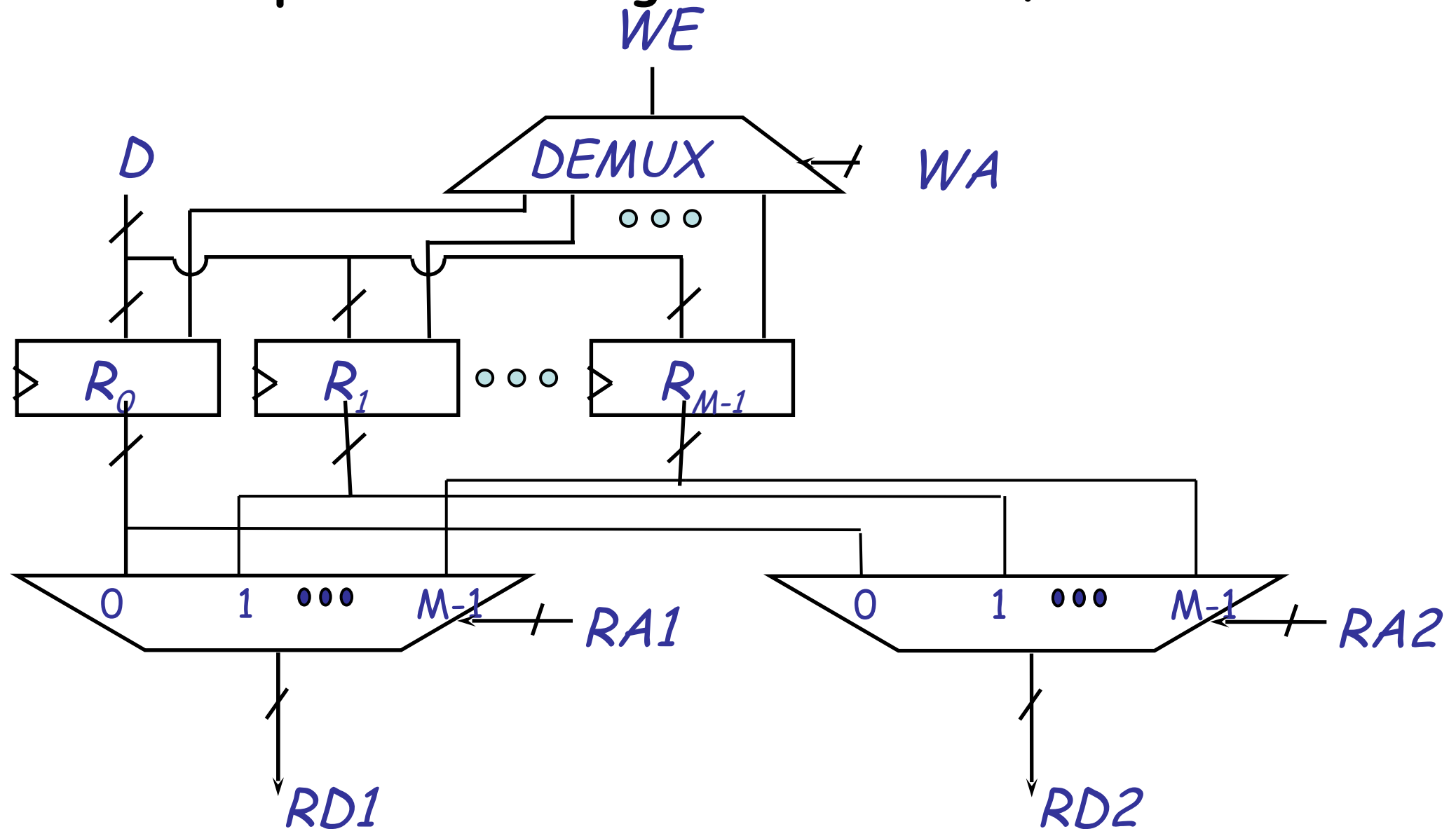


DISCS

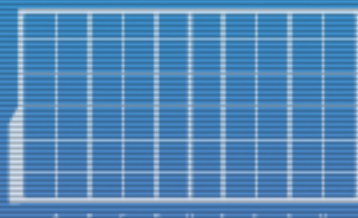
2-Port Register File

2 Combinational READ ports, 1 clocked WRITE port

- One data input (WD), two data outputs (RD1, RD2)
- There are 3 separate ADR signals now: RA1, RA2 and WA



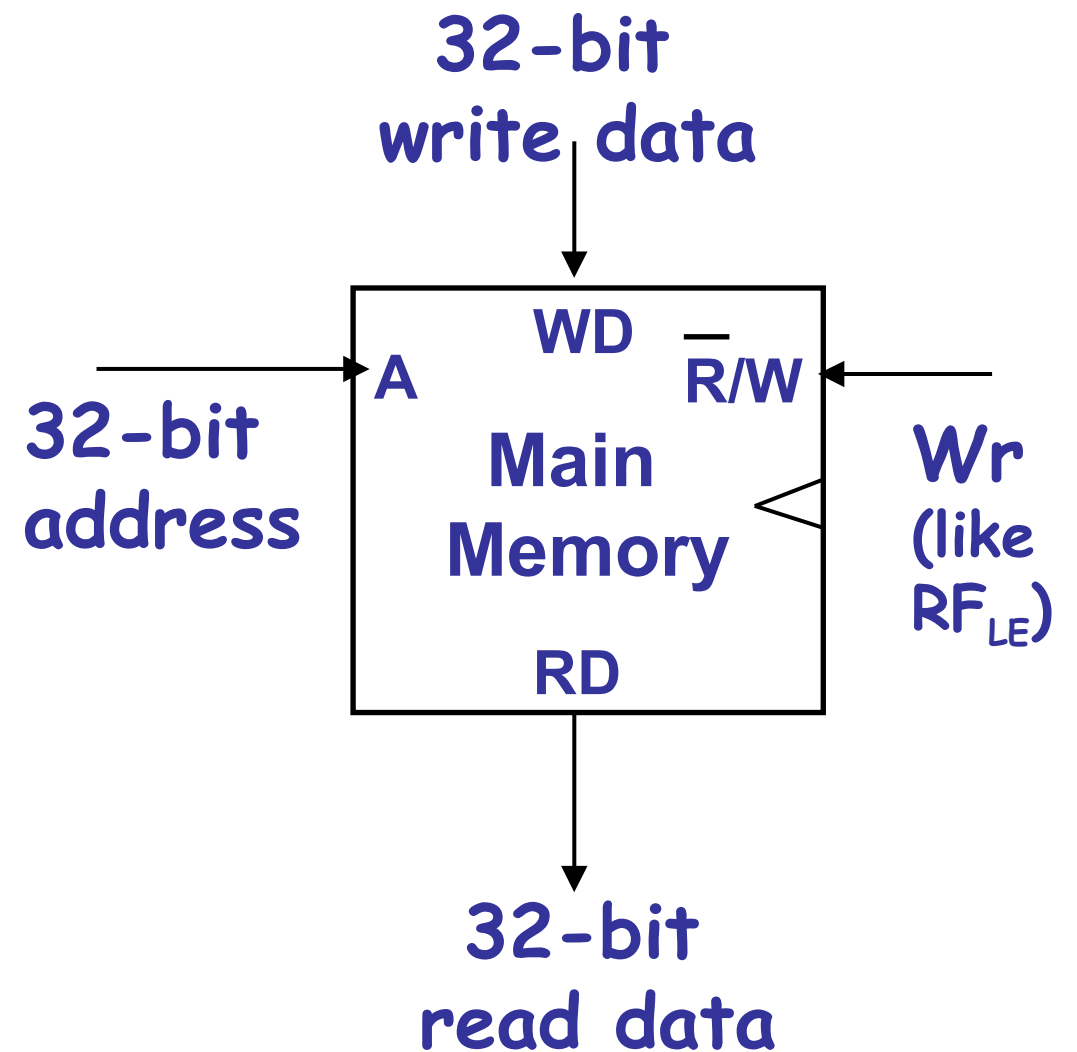
00101010010101000011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
10010100101010010100101010010101



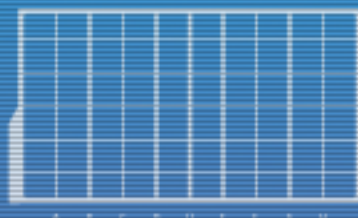
DISCS

Main Memory

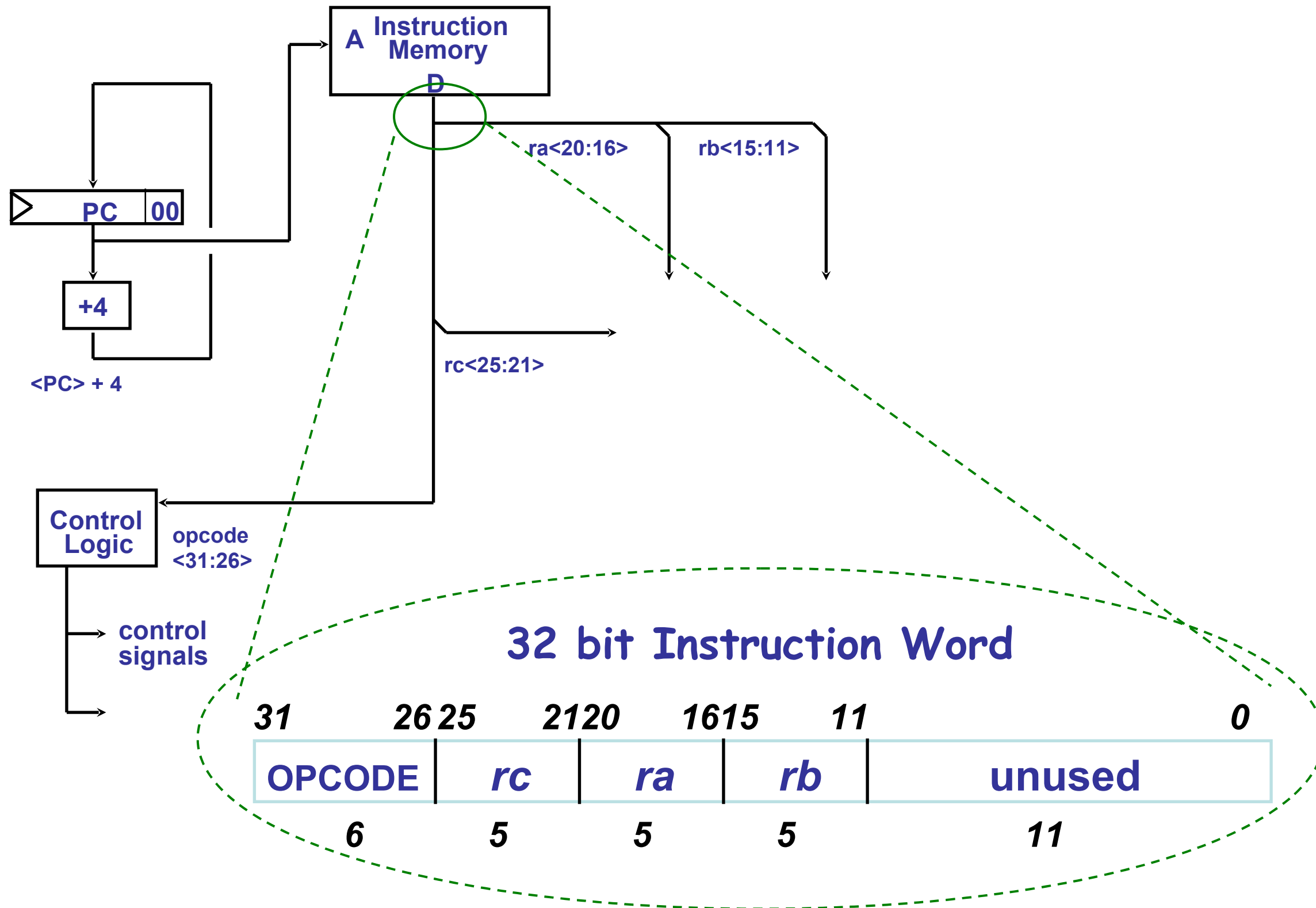
- ▶ Signals are similar to original Reg File.
 - ▶ ADR (A) signal is bigger but is shared by read and write.
 - ▶ Combinational read.
 - ▶ Clocked write.
- ▶ Memory has two “copies”:
 - ▶ Instruction memory.
 - ▶ Data memory.
- ▶ aka “Harvard Architecture” after the Harvard Mark-1.
- ▶ In reality, we have only 1 main memory, but use it for both instructions and data with separate caches.
 - ▶ So looks like Harvard anyway.



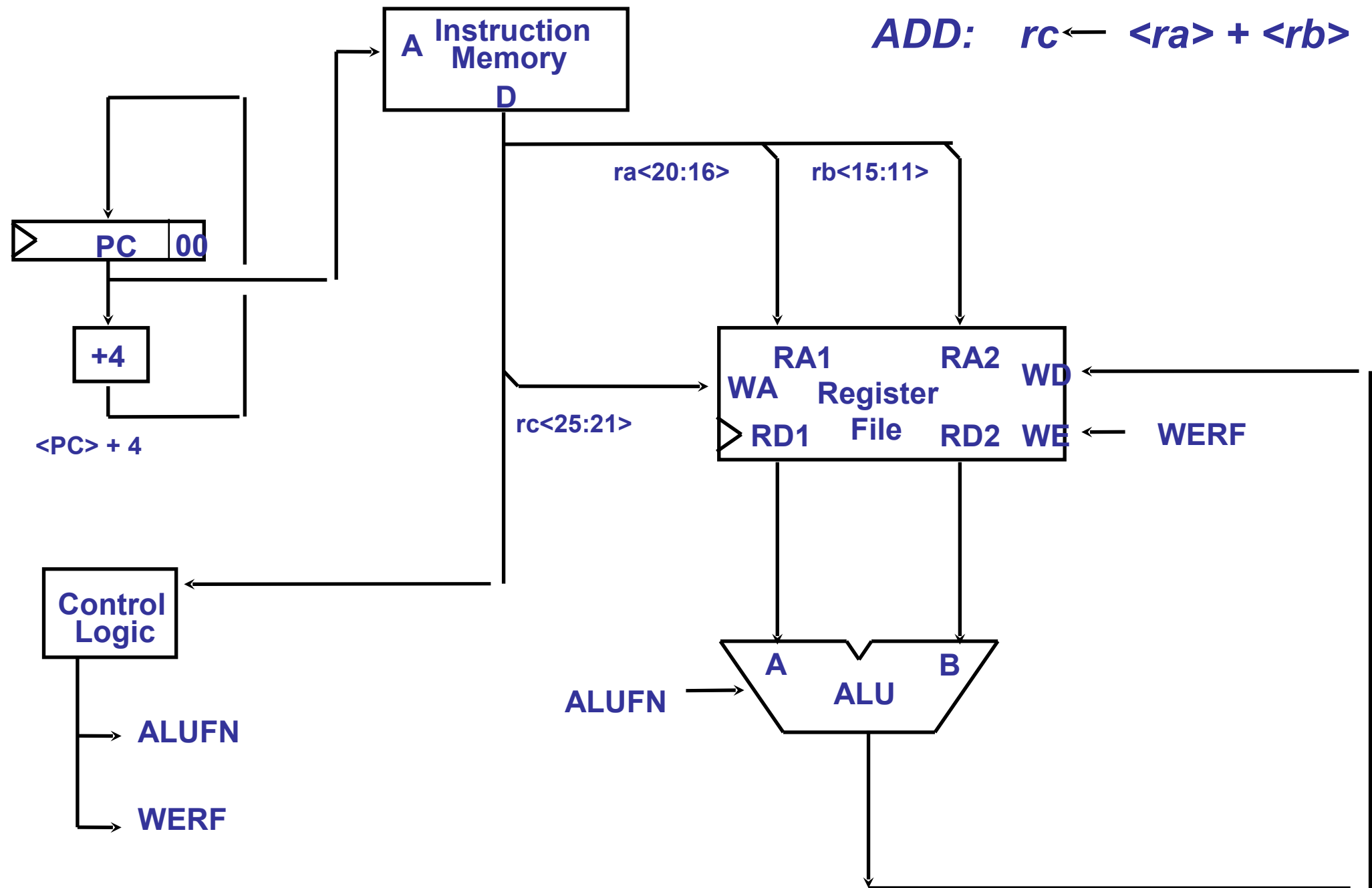
00101010010101000011110100001100
10001100100001111001101010010101
11001010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
10010101010101010101010101010101



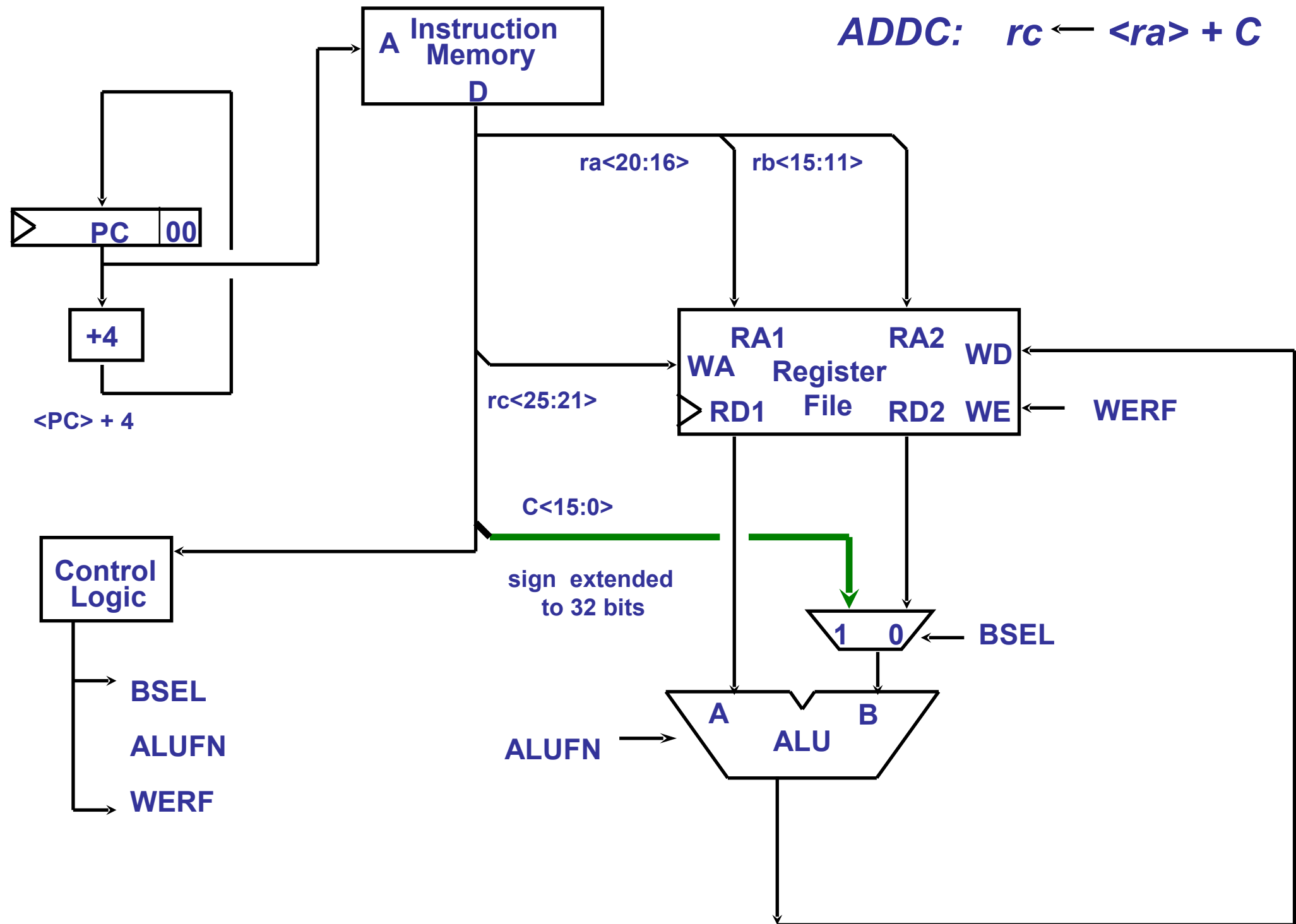
Instruction Fetch



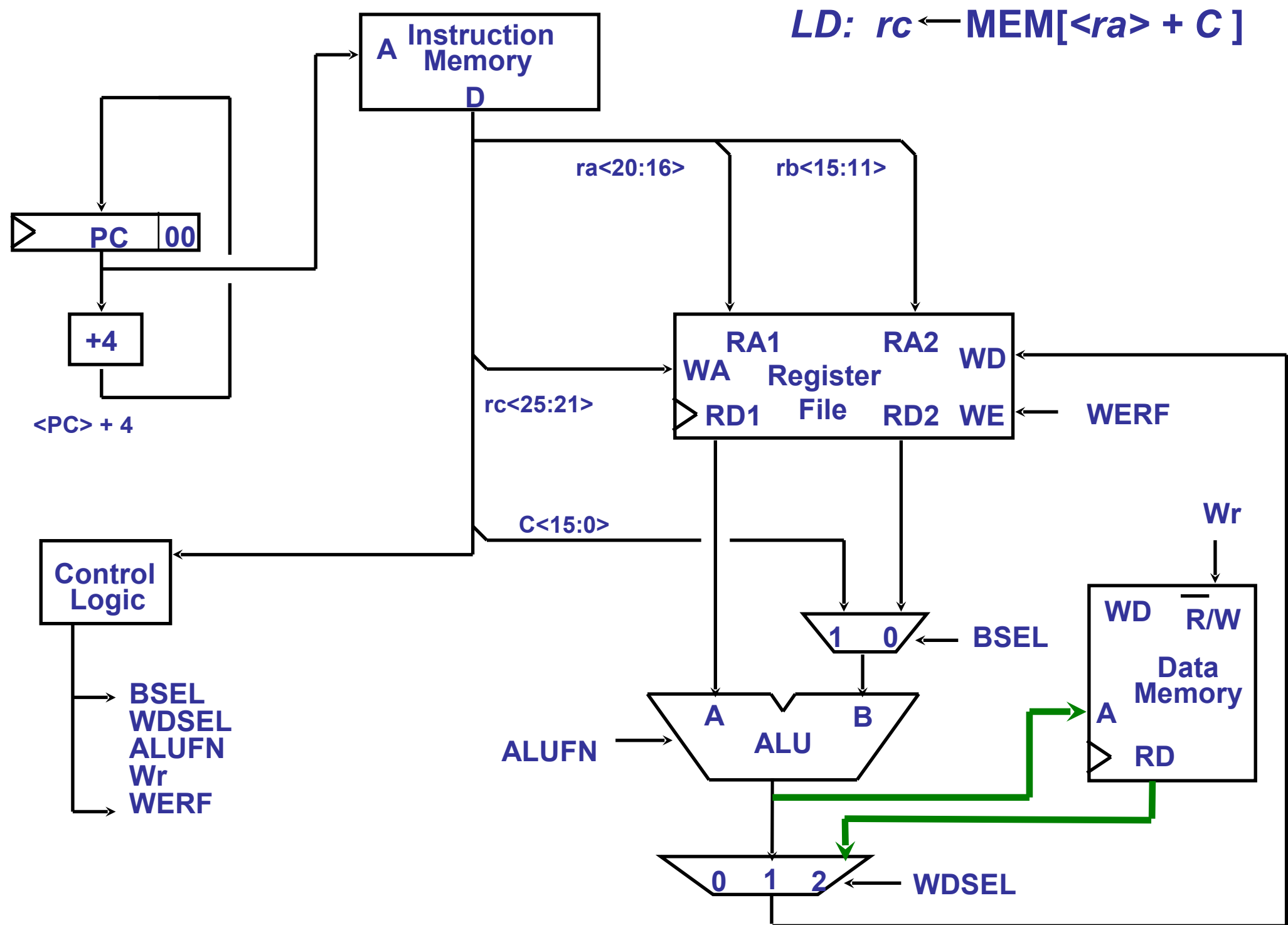
ALU Register-Register Operations



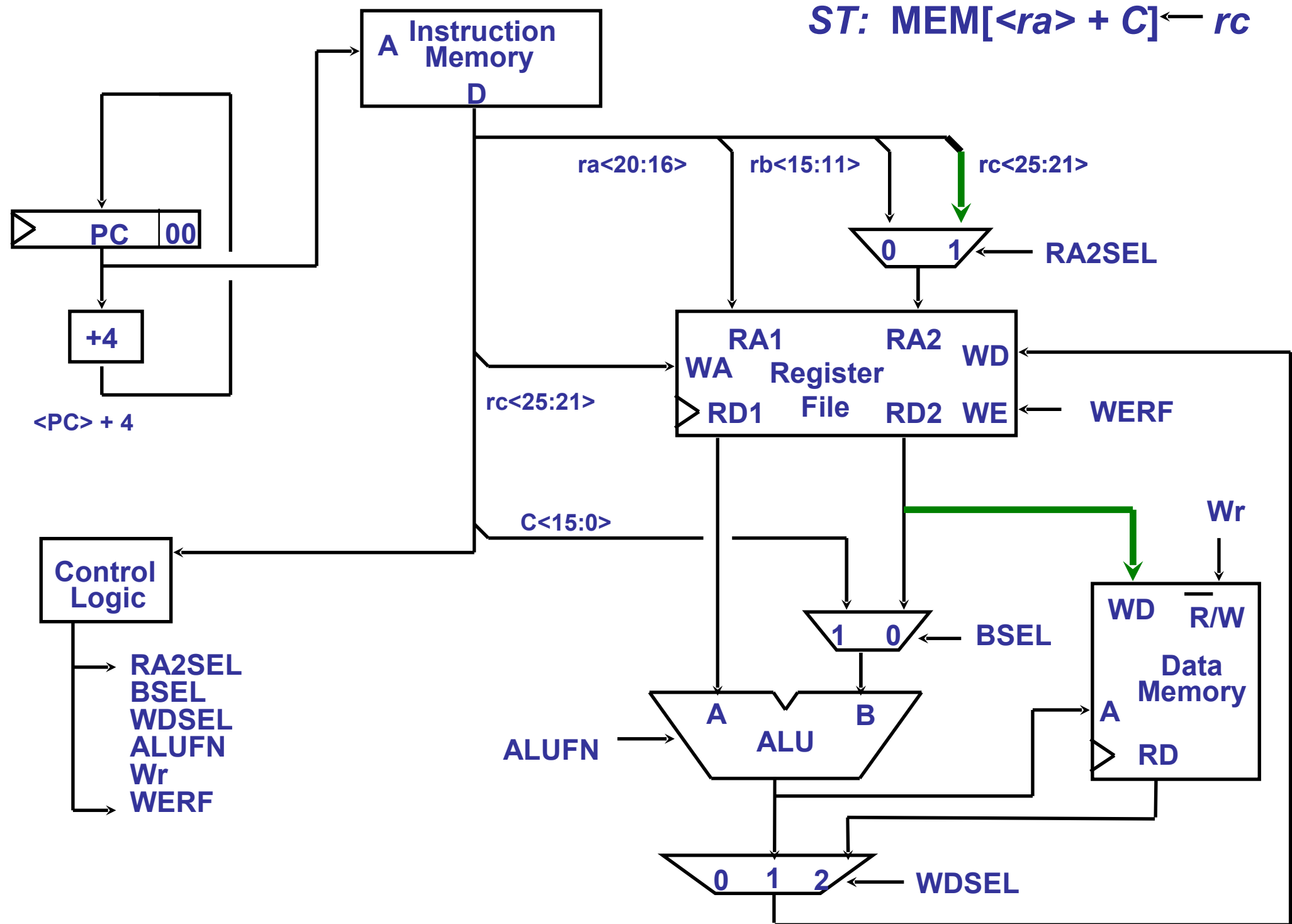
ALU Register-Constant Operations



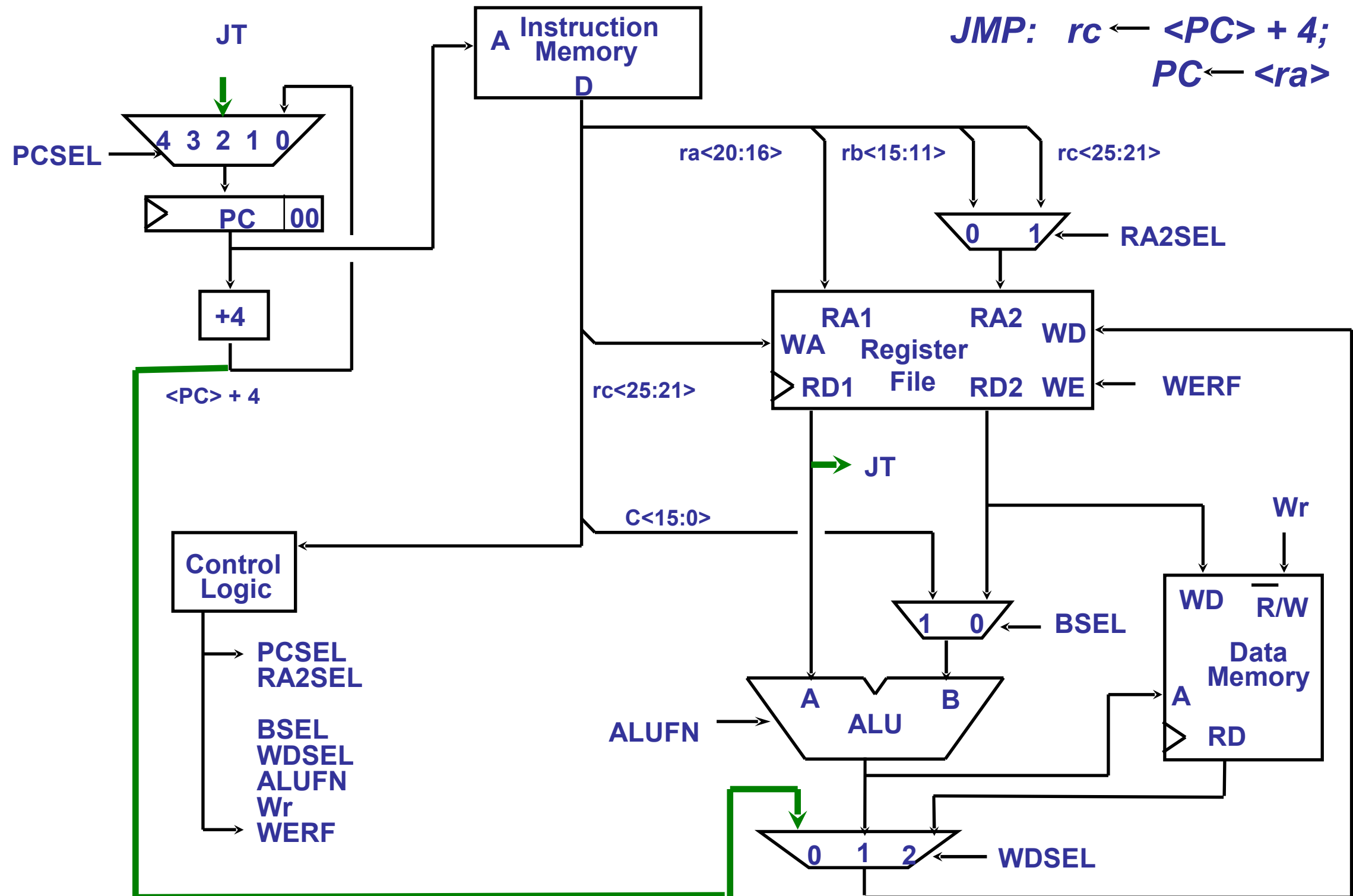
Load Instruction



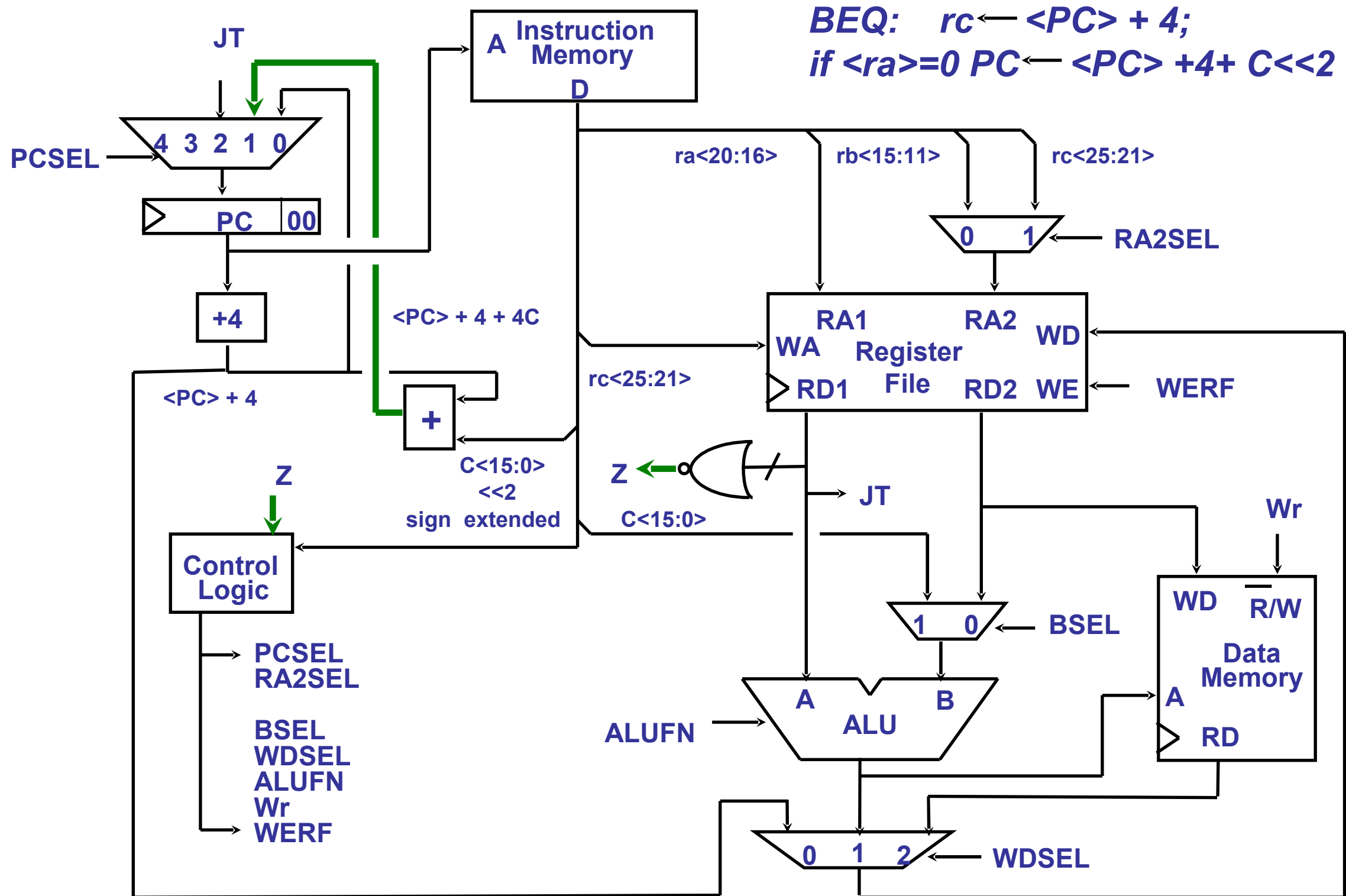
Store Instruction



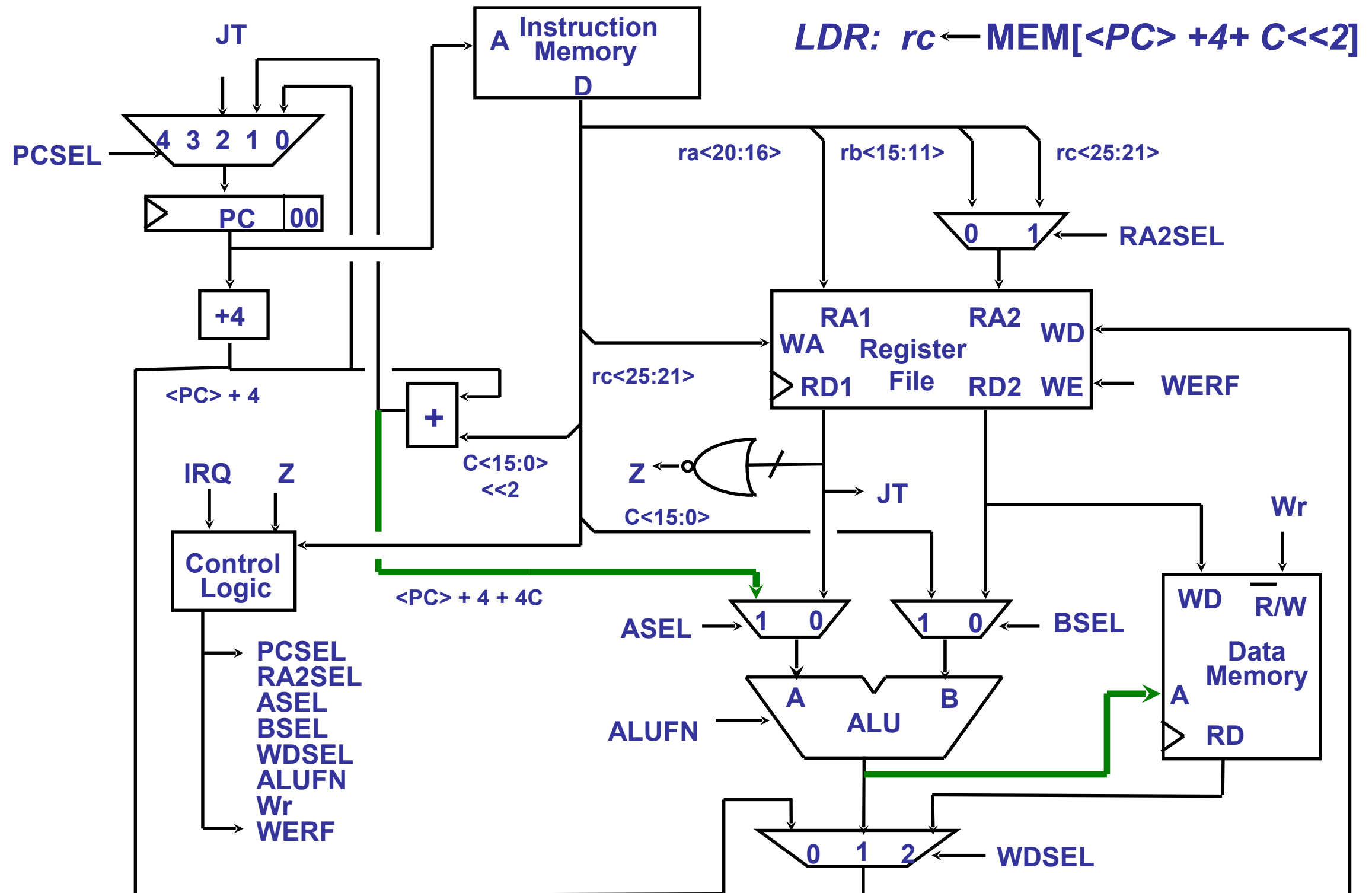
Jump Instruction



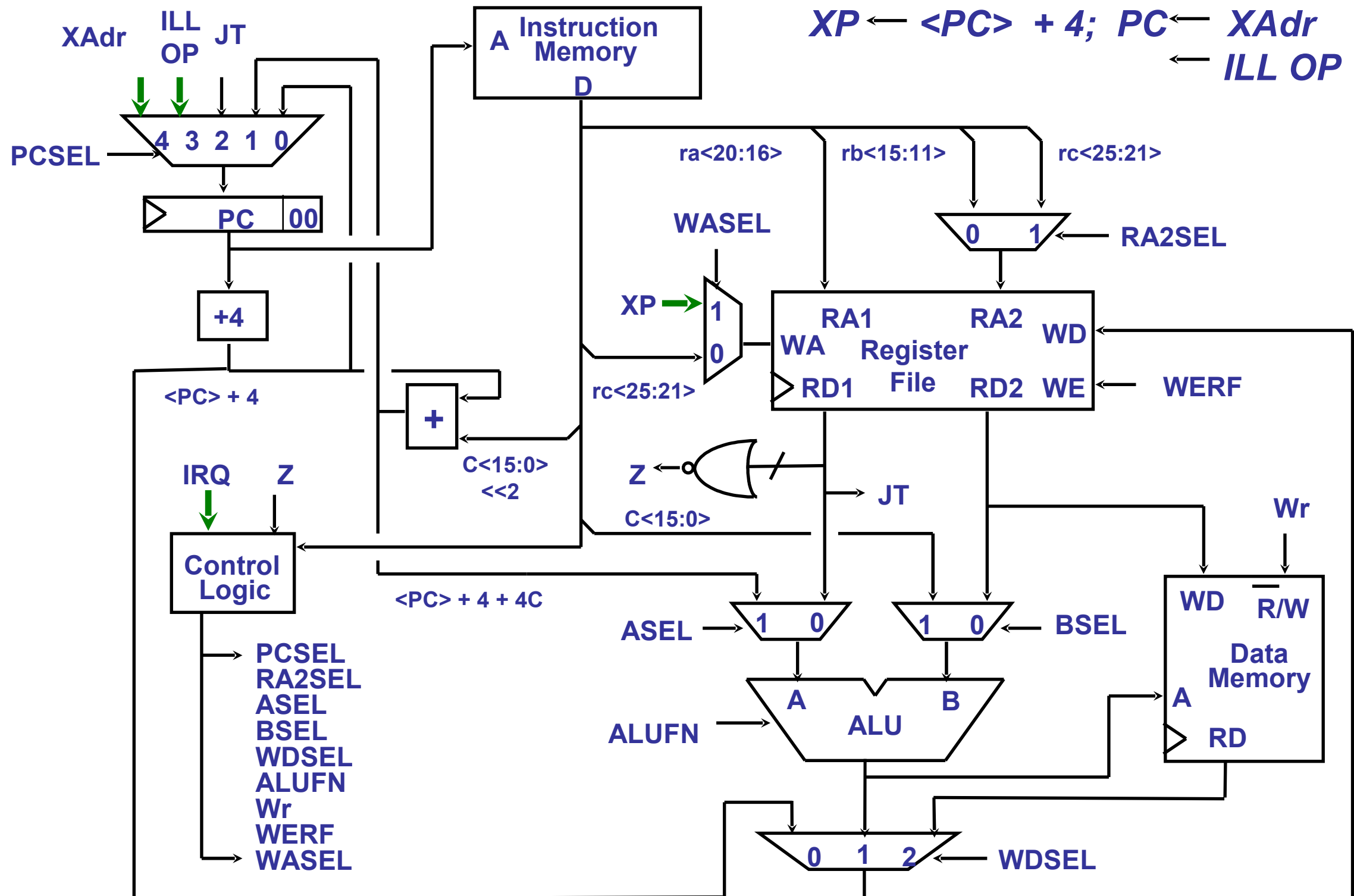
Branch Instructions (Final Lab)



Load Relative Instruction



Exceptions: Traps, Faults and Interrupts

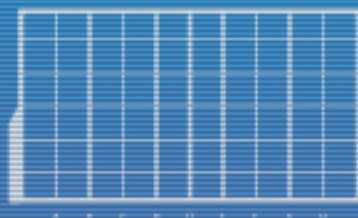


Exceptions and Interrupts

- ▶ **What's XP for? Handling “INTERRUPTS” aka “Exceptions”**
 - ▶ **FAULTS**
 - ▶ CPU or SYSTEM generated (synchronous)
 - ▶ e.g., Illegal Instruction
 - ▶ **TRAPS & system calls**
 - ▶ CPU generated (synchronous)
 - ▶ e.g., read-a-character
 - ▶ **I/O events (IRQ)**
 - ▶ externally generated (asynchronous)
 - ▶ e.g., key struck
- ▶ **How Interrupts Work:**
 - ▶ Interrupt running program.
 - ▶ Invoke exception handler (like a procedure call).
 - ▶ Return to continue execution.
- ▶ **NOT IMPORTANT in CS152 (not implemented in lab either)**

KEY:
TRANSPARENCY
to interrupted
program.
(Most difficult
for asynchronous
interrupts.)

00101010010101000011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
1001010010010010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
1001010010101001010101001010101



Exceptions and Interrupts

► IMPLEMENTATION:

- “FORCED” procedure call: new PC, save old PC for return.
- New PC comes from:
 - IllOp handler address, in case of traps and faults.
 - XAdr, in case of interrupts (IRQ=1).
- Save old PC in XP (R30).
- Don't execute current instruction in hardware.

Example: DIV unimplemented

```
LD (R31 , A , R0)
LD (R31 , B , R1)
DIV (R0 , R1 , R2)
ST (R2 , C , R31)
```

Forced by
hardware

IllOp:

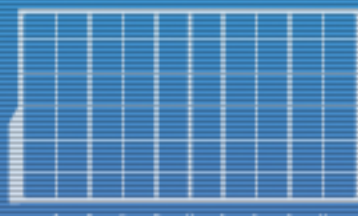
PUSH (XP)

*Fetch inst. at Mem[Reg[XP]- 4],
check for DIV opcode, get reg numbers,
perform operation in software, fill result reg*

POP (XP)

JMP (XP)

00101010010101000011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
001001010100101001001010010010110
10010100100001010100100101001010
10010100101010010100101010010101
10010101010101010101010101010101



DISCS

Specifying Control Logic Signals

Control
logic
inputs

OPCODE
Z

Signals may be
taken from
part of OPCODE

conditional

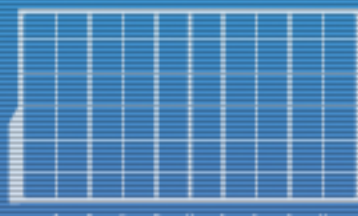
INSTR.	ADD	ADDC	LD	ST	BEQ	BNE	JMP	...
PCSEL	0				Z			
RA2SEL	0				*			
BSEL	0				*			
WDSEL	1				0			
ALUFN	op				*			
Wr	0				0			
WERF	1				1			

Control
logic
outputs

► Looking at the datapath you should be able to fill in the entire table!

► (See Final Lab for actual implementation.)

00101010010101000011110100001100
10001100100001111001101010010101
110010101010100001001100101010100
100101001001001010101010101010101
11100001111010110000000111101001
00100101010010100100100100100110
10010100100001010100100101001010
10010100101010010100101010010101
10010100101010010100101010010101



DISCS

Cheat Sheet #2: Unpipelined β Datapath

