University Of Diyala
College Of Engineering
Department of Computer Engineering

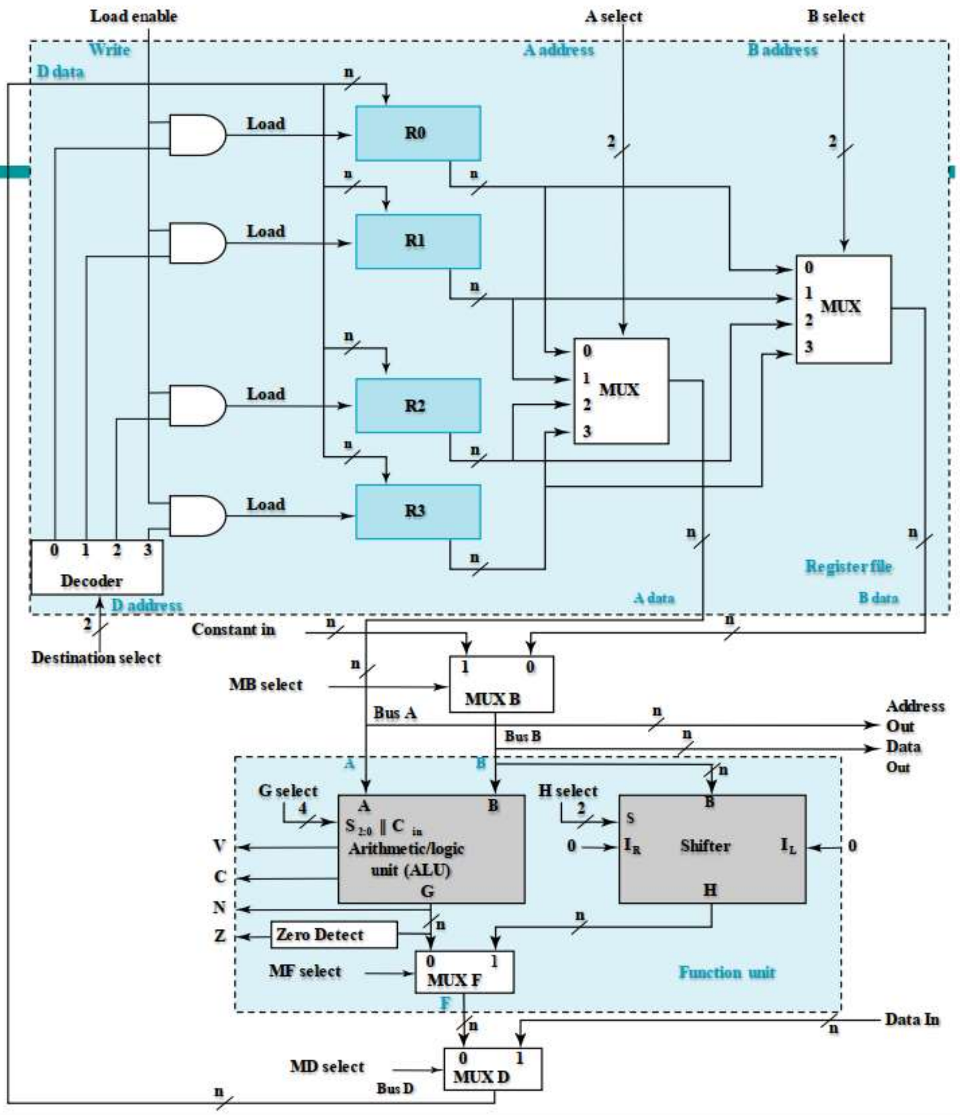
Digital System Design II Microprogramming II-Part2

Dr. Yasir Al-Zubaidi

Third stage

2021

Datapath


Load enable
A select
B select
Write
A address
B address
D data
Load
R0
R1
R2
R3
MUX
0
1
2
3
n
Decoder
D address
Destination select
Register file
A data
B data
Constant in
MB select
MUX B
Bus A
Bus B
Address Out
Data Out
G select
4
A
B
$S_{2:0} \parallel C_{in}$
Arithmetic/logic unit (ALU)
G
H select
S
I_R
Shifter
I_L
H
V
C
N
Z
Zero Detect
MF select
MUX F
F
Function unit
Data In
MD select
MUX D
Bus D

Instruction Set Architecture (ISA) for Simple Computer (SC)

- Instructions are stored in RAM or ROM as a *program,* the addresses for instructions are provided by a *program counter (PC)*
  - Count up or load a new address
  - The PC and associated control logic are part of the Control Unit

- A typical instruction specifies:
  - Operands to use
  - Operation to be performed
  - Where to place the result, or which instruction to execute next

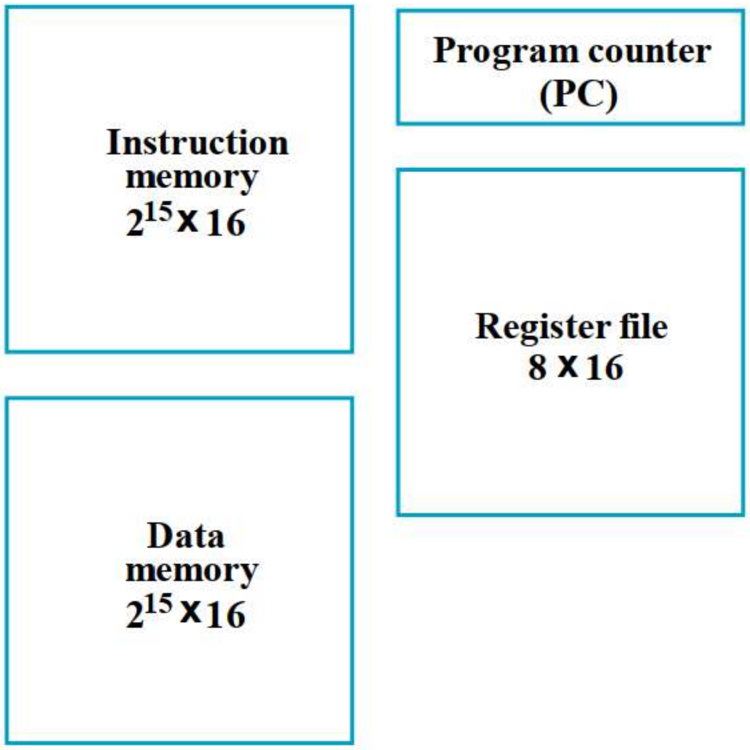
- Executing an instruction
  - Activate the necessary sequence of operations specified by the instruction
  - Be controlled by the control unit and performed in:
    - datapath
    - control unit
    - external hardware such as memory or input/output

Example ISAs

- RISC (Reduced Instruction Set Computer)
  - Digital Alpha
  - Sun Sparc
  - MIPS RX000
  - IBM PowerPC
  - HP PA/RISC
- CISC (Complex Instruction Set Computer)
  - Intel x86
  - Motorola 68000
  - DEC VAX
- VLIW (Very Large Instruction Word)
  - Intel Itanium

ISA: Storage Resources

- "Harvard architecture“: Separate instruction and data memories
- Permit use of single clock cycle per instruction implementation
- Due to use of "cache" in modern computer architectures, it is a fairly realistic model


Instruction memory $2^{15} \times 16$
Data memory $2^{15} \times 16$
Program counter (PC)
Register file 8×16

ISA: Instruction Format

- The three formats are: Register, Immediate, and Jump/Branch

15 9	8 6	5 3	2 0
Opcode	Destination register (DR)	Source register A (SA)	Source register B (SB)

(a) Register

15 9	8 6	5 3	2 0
Opcode	Destination register (DR)	Source register A (SA)	Operand (OP)

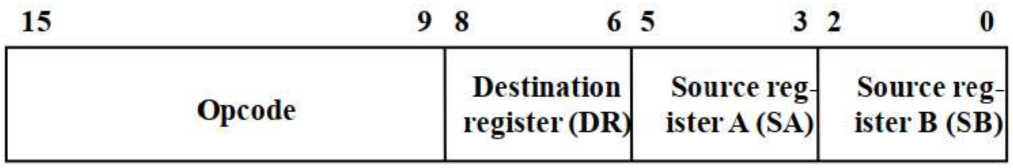
(b) Immediate

15 9	8 6	5 3	2 0
Opcode	Address (AD) (Left)	Source register A (SA)	Address (AD) (Right)

(c) Jump and Branch

- All formats contain an Opcode field in bits 9 through 15.
  - The Opcode specifies the operation to be performed

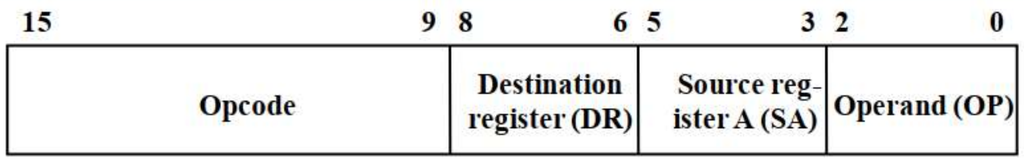
ISA: Instruction Format - Register



(a) Register

- This format supports:
  - R1 ← R2 + R3
  - R1 ← sl R2
- Three 3-bit register fields:
  - DR - destination register (R1 in the examples)
  - SA - the A source register (R2 in the first example)
  - SB - the B source register (R3 in the first example and R2 in the second example)
- Why is R2 in the second example SB instead of SA?

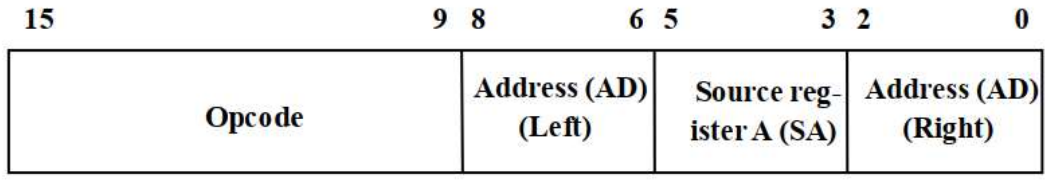
ISA: Instruction Format - Immediate



(b) Immediate

- This format supports:
  - R1 ← R2 + 3
- The B Source Register field is replaced by an Operand field OP specifying a constant. (3-bit constant, values from 0 to 7)
- The constant:
  - Zero-fill (on the left of) the operand to form 16-bit constant
  - 16-bit representation for values 0 through 7

ISA: Instruction Format – Jump & Branch



(c) Jump and Branch

- This instruction supports changes in the sequence of instruction execution by adding an extended, 6-bit, signed 2's-complement *address offset* to the PC value

- The SA field: permits jumps and branches on N or Z based on the contents of *Source register A*

- The Address (AD) field (6-bit) replaces the DR and SB fields
  - Example: Suppose that a jump for the Opcode and the PC contains 45 (0…0101101) and AD contains – 12 (110100). Then the new PC value will be:
    0…0101101 + (1…110100) = 0…0100001   (i.e., 45 + (– 12) = 33)

ISA: Instruction Specifications

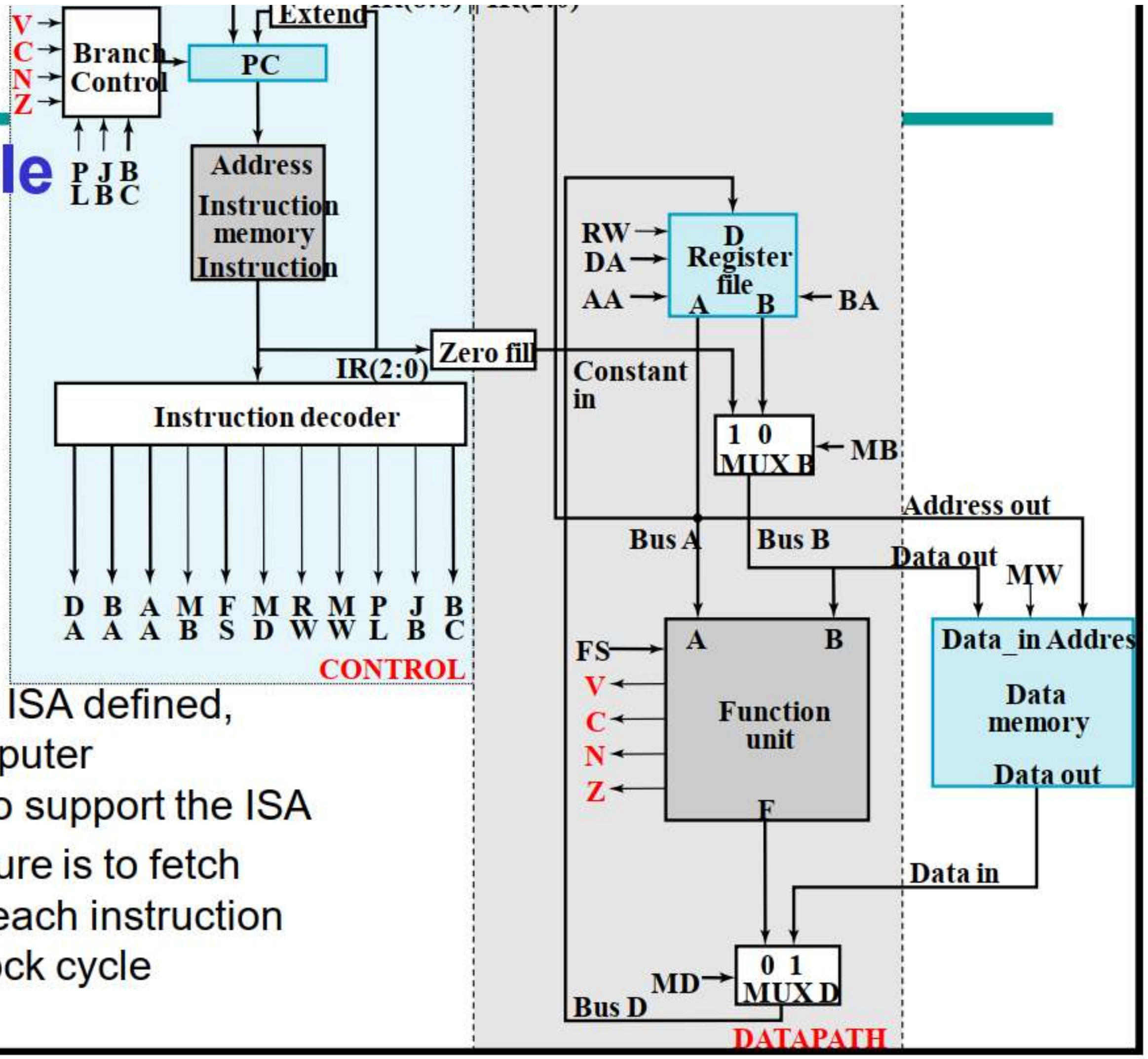
Instruction	Opcode	Mnemonic	Format	Description	Status Bits
Move A	0000000	MOVA	RD,RA	$R[DR] \leftarrow R[SA]$	N, Z
Increment	0000001	INC	RD,RA	$R[DR] \leftarrow R[SA] + 1$	N, Z
Add	0000010	ADD	RD,RA,RB	$R[DR] \leftarrow R[SA] + R[SB]$	N, Z
Subtract	0000101	SUB	RD,RA,RB	$R[DR] \leftarrow R[SA] - R[SB]$	N, Z
Decrement	0000110	DEC	RD,RA	$R[DR] \leftarrow R[SA] - 1$	N, Z
AND	0001000	AND	RD,RA,RB	$R[DR] \leftarrow R[SA] \wedge R[SB]$	N, Z
OR	0001001	OR	RD,RA,RB	$R[DR] \leftarrow R[SA] \vee R[SB]$	N, Z
Exclusive OR	0001010	XOR	RD,RA,RB	$R[DR] \leftarrow R[SA] \oplus R[SB]$	N, Z
NOT	0001011	NOT	RD,RA	$R[DR] \leftarrow \overline{R[SA]}$	N, Z
Move B	0001100	MOVB	RD,RB	$R[DR] \leftarrow R[SB]$	
Shift Right	0001101	SHR	RD,RB	$R[DR] \leftarrow \mathrm{sr}\ R[SB]$	
Shift Left	0001110	SHL	RD,RB	$R[DR] \leftarrow \mathrm{sl}\ R[SB]$	
Load Immediate	1001100	LDI	RD, OP	$R[DR] \leftarrow \mathrm{zf}\ OP$	
Add Immediate	1000010	ADI	RD,RA,OP	$R[DR] \leftarrow R[SA] + \mathrm{zf}\ OP$	
Load	0010000	LD	RD,RA	$R[DR] \leftarrow M[R[SA]]$	
Store	0100000	ST	RA,RB	$M[R[SA]] \leftarrow R[SB]$	
Branch on Zero	1100000	BRZ	RA,AD	$\text{if } (R[SA] = 0)\ PC \leftarrow PC + \mathrm{se}\ AD$	
Branch on Negative	1100001	BRN	RA,AD	$\text{if } (R[SA] < 0)\ PC \leftarrow PC + \mathrm{se}\ AD$	
Jump	1110000	JMP	RA	$PC \leftarrow R[SA]$	

ISA:Example Instructions and Data in Memory

Memory Representation of Instruction and Data

Decimal Address	Memory Contents	Decimal Opcode	Other Field	Operation
25	0000101 001 010 011	5 (Subtract)	DR:1, SA:2, SB:3	R1 ← R2 − R3
35	0100000 000 100 101	32 (Store)	SA:4, SB:5	M[R4] ← R5
45	1000010 010 111 011	66 (Add Im mediate)	DR: 2, SA :7, OP :3	R2 ← R7 + 3
55	1100000 101 110 100	96 (Branch on Zero)	AD: 44, SA:6	If R6 = 0, PC ← PC − 20
70	0000000 00110 0 000	Data = 192. After execution of instruction in 35, Data = 80.		

Single-Cycle Hardwired Control:



- Based on the ISA defined, design a computer architecture to support the ISA
- The architecture is to fetch and execute each instruction in a single clock cycle

The Control Unit

- Datapath: the Data Memory has been attached to the *Address Out*, *Data Out,* and *Data In* lines of the Datapath.

- Control Unit:
  - The MW input to the Data Memory is the Memory Write signal from the Control Unit.
  - The Instruction Memory address input is provided by the PC and its instruction output feeds the Instruction Decoder.
  - Zero-filled IR(2:0) becomes Constant In
  - Extended IR(8:6) || IR(2:0) and Bus A are address inputs to the PC.
  - The PC is controlled by Branch Control logic

Program Counter (PC) Function

- PC function is based on instruction specifications involving jumps and branches:

Branch on Zero	**BRZ**	**if (R[SA] = 0) PC $\leftarrow$ PC + seAD**
Branch on Negative	**BRN**	**if (R[SA] < 0) PC $\leftarrow$ PC + seAD**
Jump	**JMP**	**PC $\leftarrow$ R[SA]**

  - The first two transfers require addition to the PC of:
    - Address Offset = Extended IR(8:6) || IR(2:0)
  - The third transfer requires that the PC be loaded with:
    - Jump Address = Bus A = R[SA]

- In addition to the above register transfers, the PC must implement the counting function:
  - PC $\leftarrow$ PC + 1

PC Function (Contd.)

- Branch Control determines the PC transfers based on five inputs:
  - N,Z – negative and zero status bits
  - PL – load enable for the PC
  - JB – Jump/Branch select: If JB = 1, Jump, else Branch
  - BC – Branch Condition select: If BC = 1, branch for N = 1, else branch for Z = 1.

PL	JB	BC	PC Operation
0	X	X	Count Up
1	1	X	Jump
1	0	1	Branch on Negative (else Count Up)
1	0	0	Branch on Zero (else Count Up)

Instruction Decoder

- Converts the instruction into the signals necessary to control the computer during the single cycle execution, combinational
  - Inputs: the 16-bit Instruction
  - Outputs: control signals
    - DA, AA, and BA: Register file addresses (IR (8:0))
      - simply pass-through signals: DA = DR, AA = SA, and BA = SB
    - FS: Function Unit Select
    - MB and MD: Multiplexer Select Controls
    - RW and MW: Register file and Data Memory Write Controls
    - PL, JB, and BC: PC Controls
- Observe that for other than branches and jumps, FS = IR(12:9)
  - The other control signals should depend as much as possible on IR(15:13)

Instruction Decoder (Contd.)

Truth Table for Instruction Decoder Logic

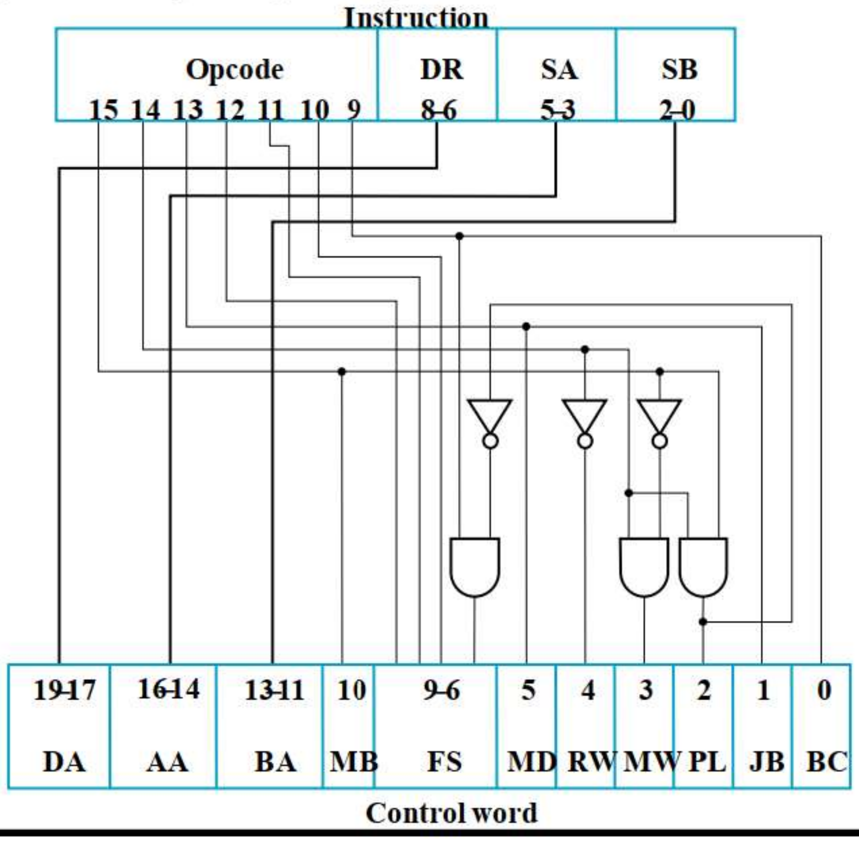
	Instruction Bits				Control Word Bits						
Instruction Function Type	**15**	**14**	**13**	**9**	**MB**	**MD**	**RW**	**MW**	**PL**	**JB**	**BC**
1. Function unit operations using registers	0	0	0	X	0	0	1	0	0	X	X
2. Memory read	0	0	1	X	0	1	1	0	0	X	X
3. Memory write	0	1	0	X	0	X	0	1	0	X	X
4. Function unit operations using register and constant	1	0	0	X	1	0	1	0	0	X	X
5. Conditional branch on zero (Z)	1	1	0	0	X	X	0	0	1	0	0
6. Conditional branch on negative (N)	1	1	0	1	X	X	0	0	1	0	1
7. Unconditional Jump	1	1	1	X	X	X	0	0	1	1	X

Instruction Decoder (Contd.)

- Instruction types are based on the control blocks and the seven control signals to be generated (MB, MD, RW, MW, PL, JB, BC):
  - Datapath and Memory Control (types 1-4)
    - Mux B
    - Memory and Mux D
  - PC Control (types 5-7)
    - Bit 15 = Bit 14 = 1 => PL
    - Bit 13 => JB.
    - Bit 9 was use as BC which contradicts FS = 0000 needed for branches. To force FS(0) to 0 for branches, Bit 9 into FS(0) is disabled by PL.

Instruction Decoder (Contd.)

- The end result by use of the types, careful assignment of codes, and use of don't cares, yields very simple logic:
- This completes the design of most of the essential parts of the single-cycle simple computer



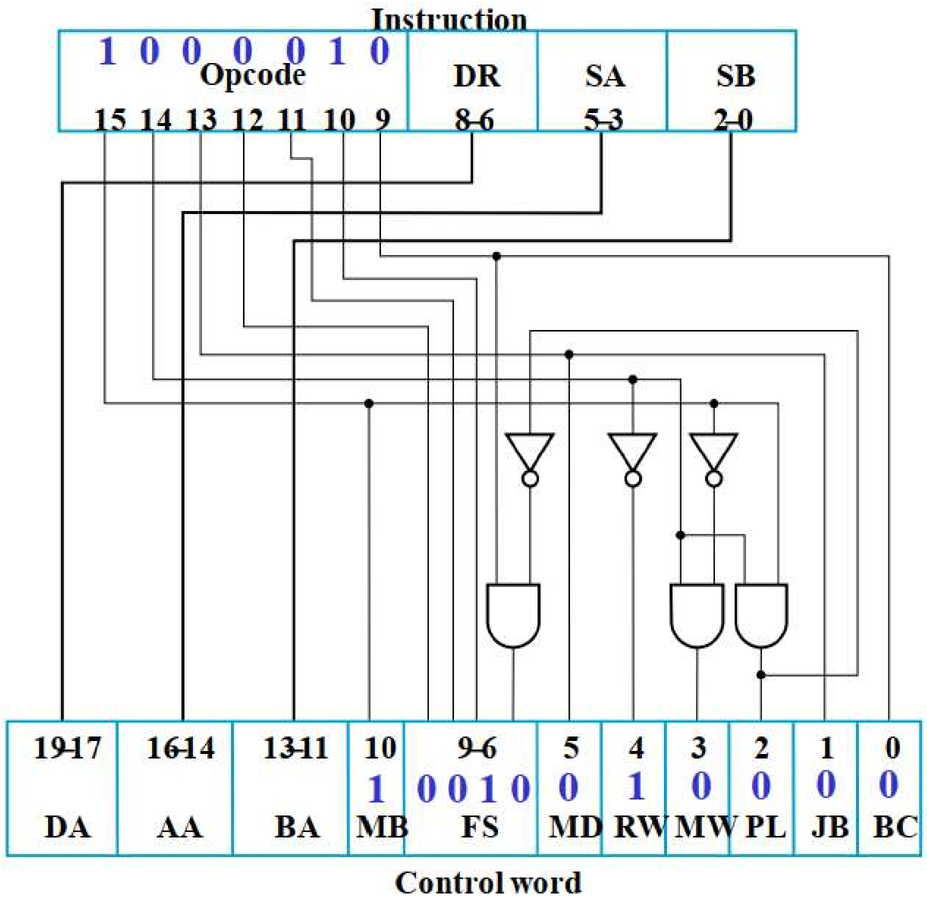
Example Instruction Execution

Six Instructions for the Single-Cycle Computer

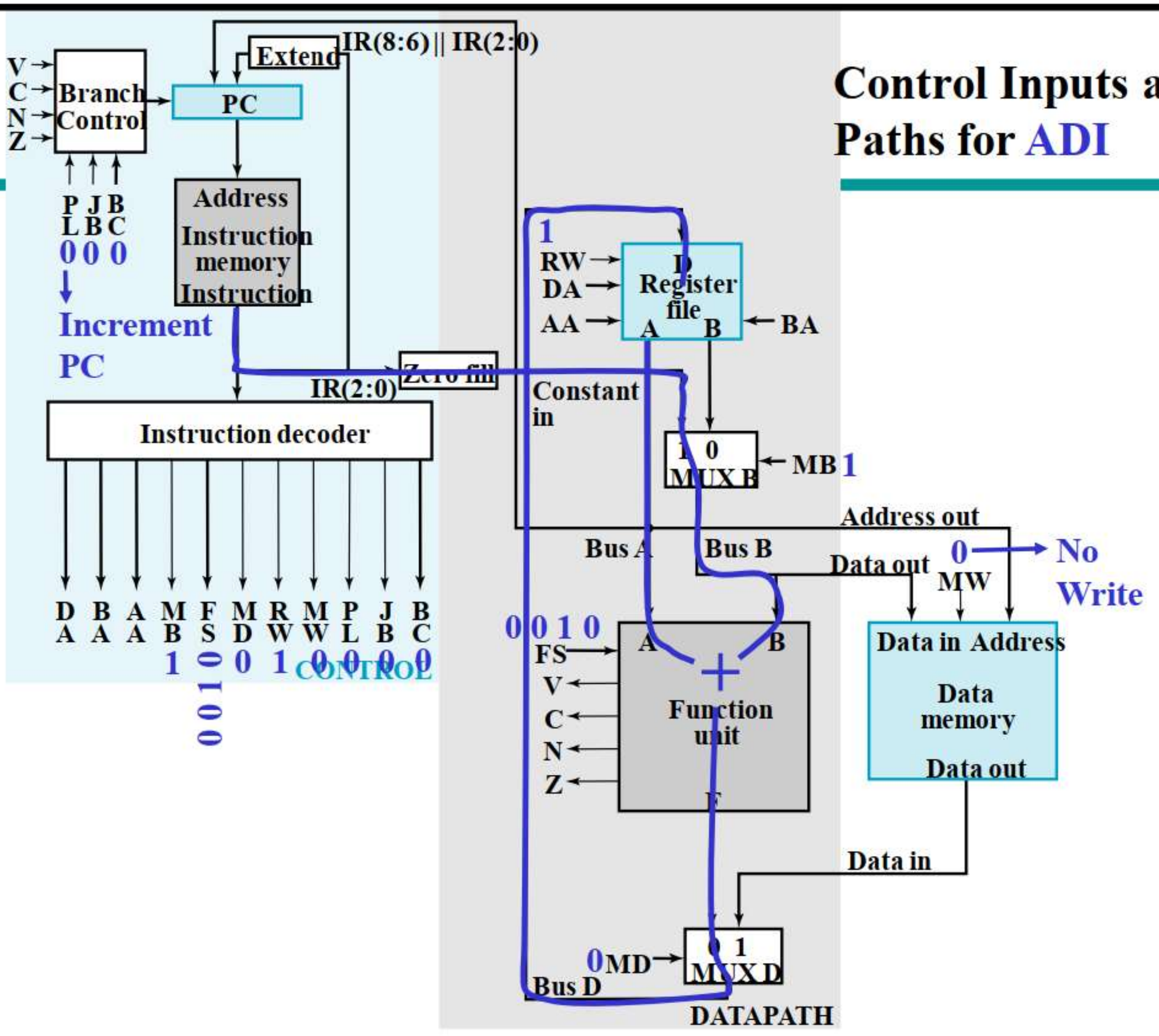
Operation code	Symbolic name	Format	Description	Function	MB	MD	RW	MW	PL	JB	BC
1000 010	ADI	Immediate	Add immediate operand	$R[DR] \leftarrow R[SA] + \text{zf } I(2{:}0)$	1	0	1	0	0	0	0
0010 000	LD	Register	Load memory content into register	$R[DR] \leftarrow M[R[SA]]$	0	1	1	0	0	1	0
0100 000	ST	Register	Store register content in memory	$M[R[SA]] \leftarrow R[SB]$	0	1	0	1	0	0	0
0001 110	SL	Register	Shift left	$R[DR] \leftarrow \text{sl } R[SB]$	0	0	1	0	0	1	0
0001 011	NOT	Register	Complement register	$R[DR] \leftarrow \overline{R[SA]}$	0	0	1	0	0	0	1
1100 000	BRZ	Jump/Branch	If $R[SA] = 0$, branch to PC + se AD	If $R[SA] = 0$, $PC \leftarrow PC + \text{se } AD$, If $R[SA] \neq 0$, $PC \leftarrow PC + 1$	1	0	0	0	1	0	0

- Decoding, control inputs and paths shown for ADI, LD and BRZ on next 6 slides

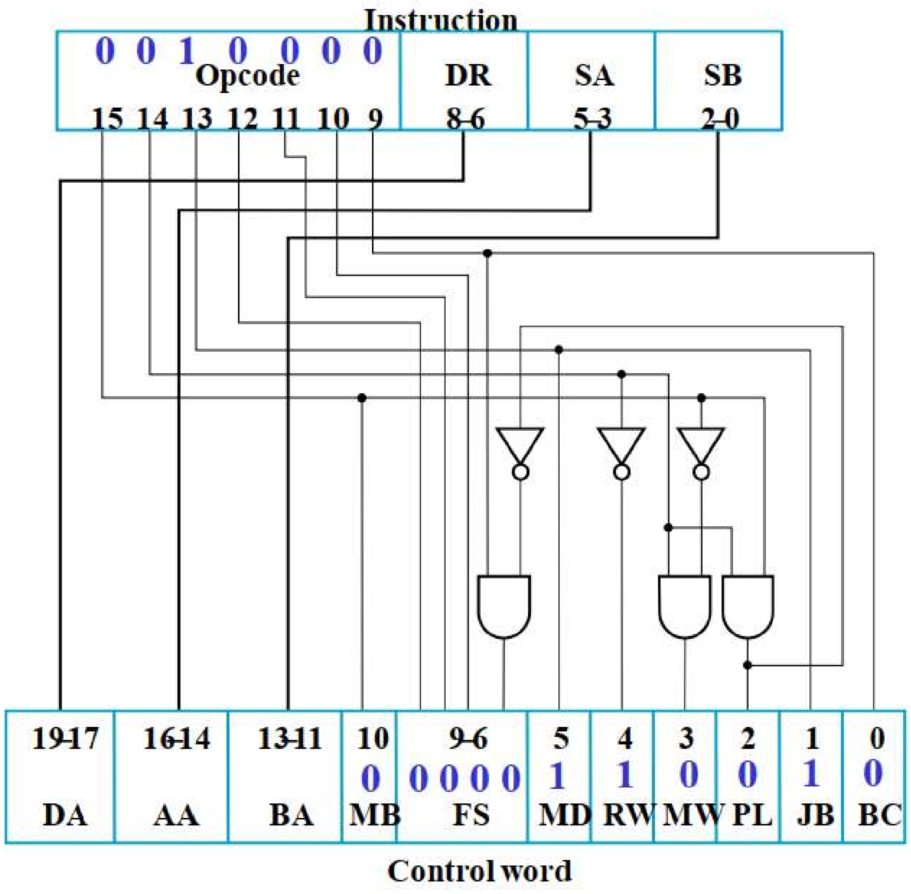
Decoding for ADI



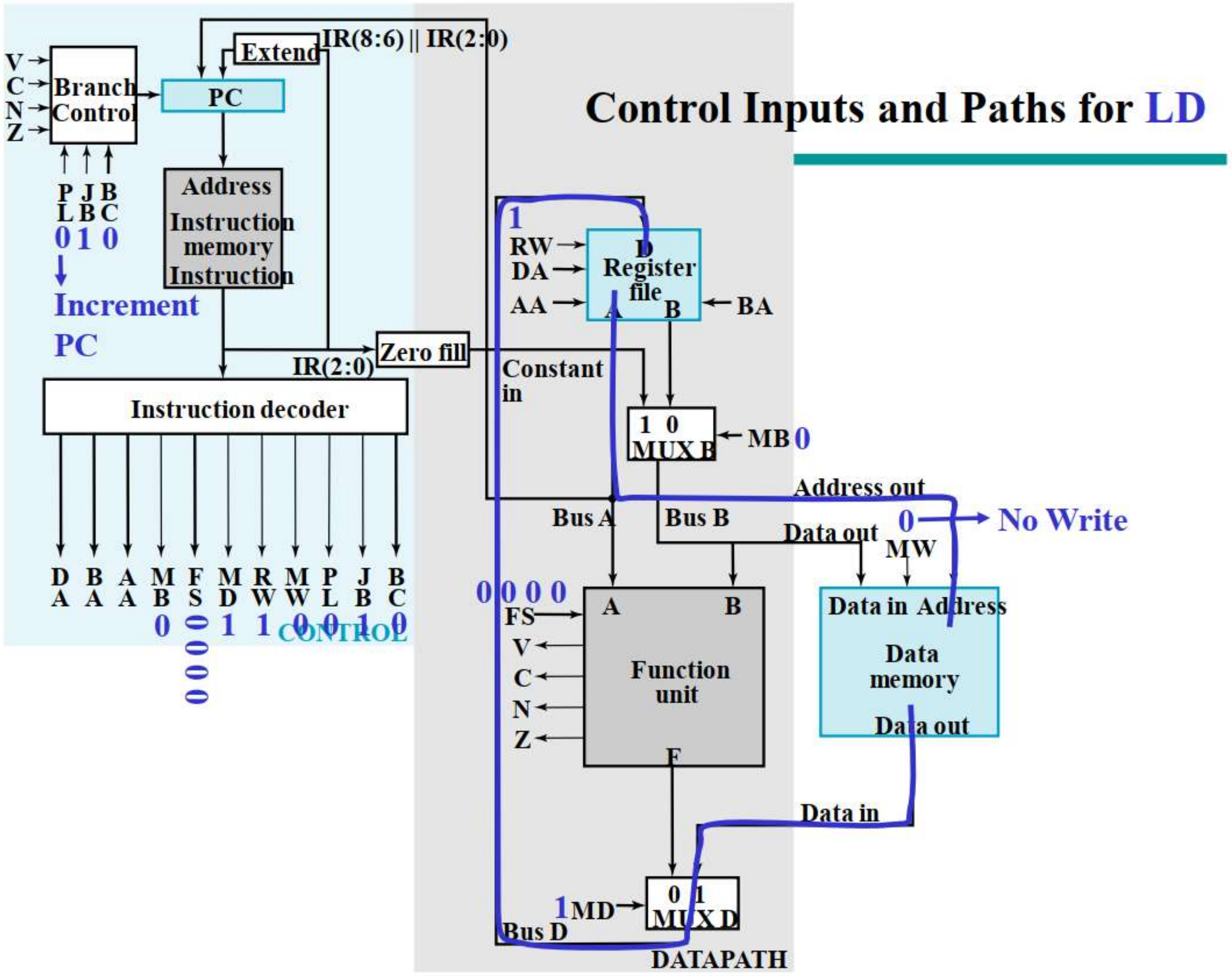
Control Inputs and Paths for ADI



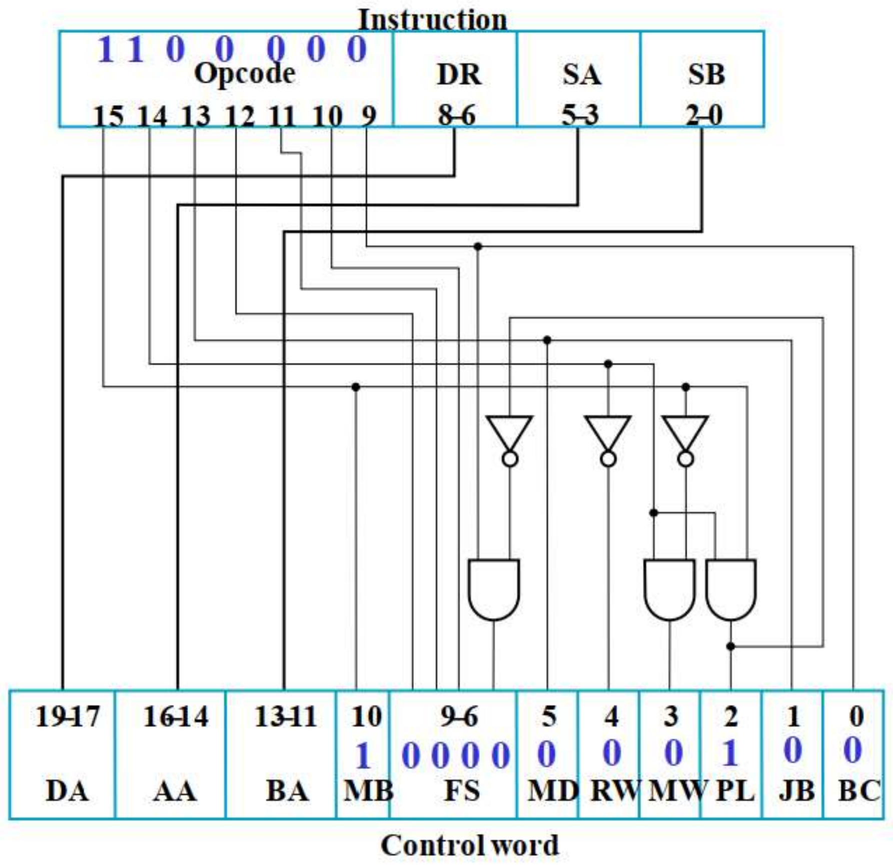
Decoding for LD



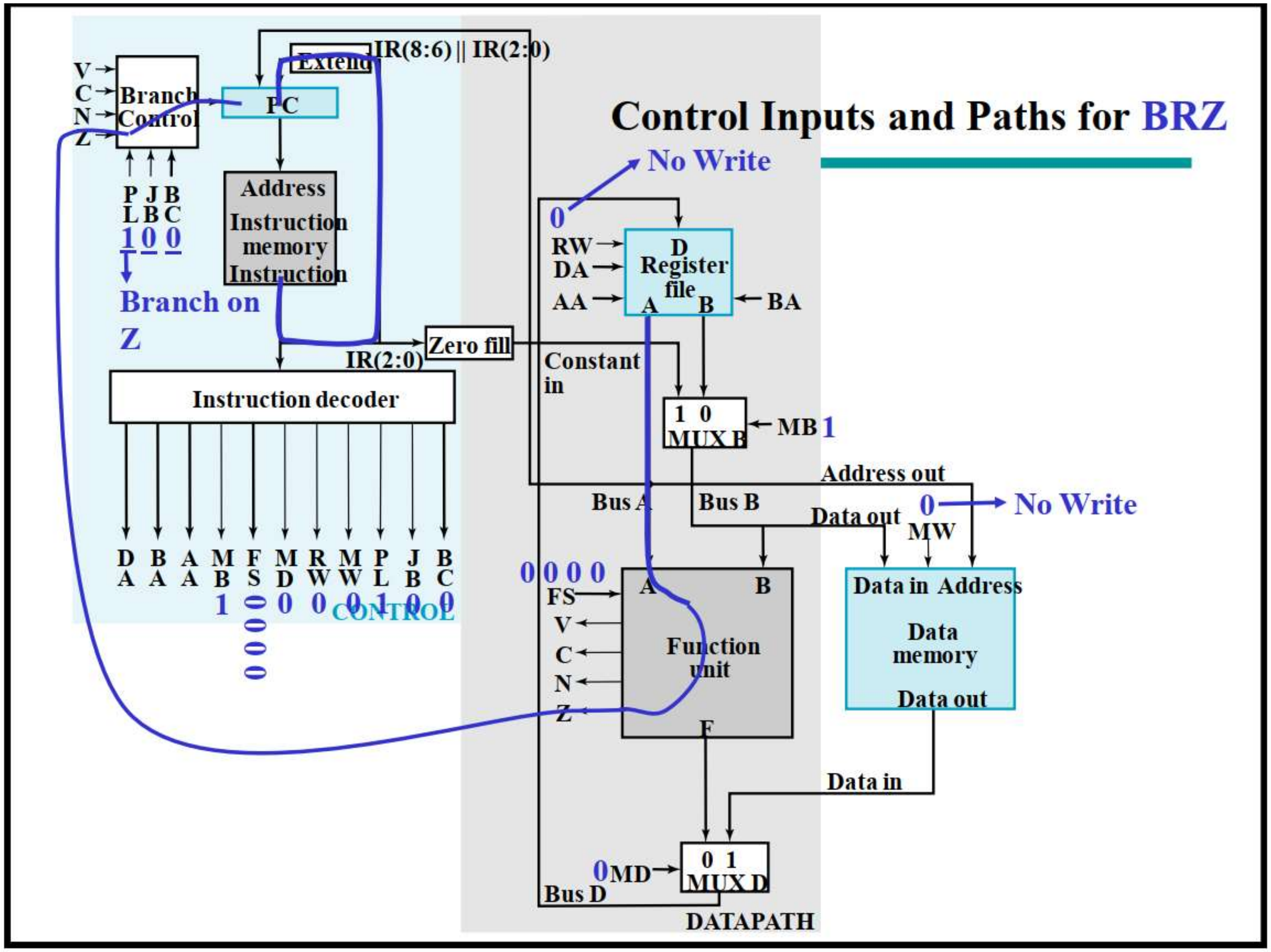
Control Inputs and Paths for LD



Decoding for BRZ

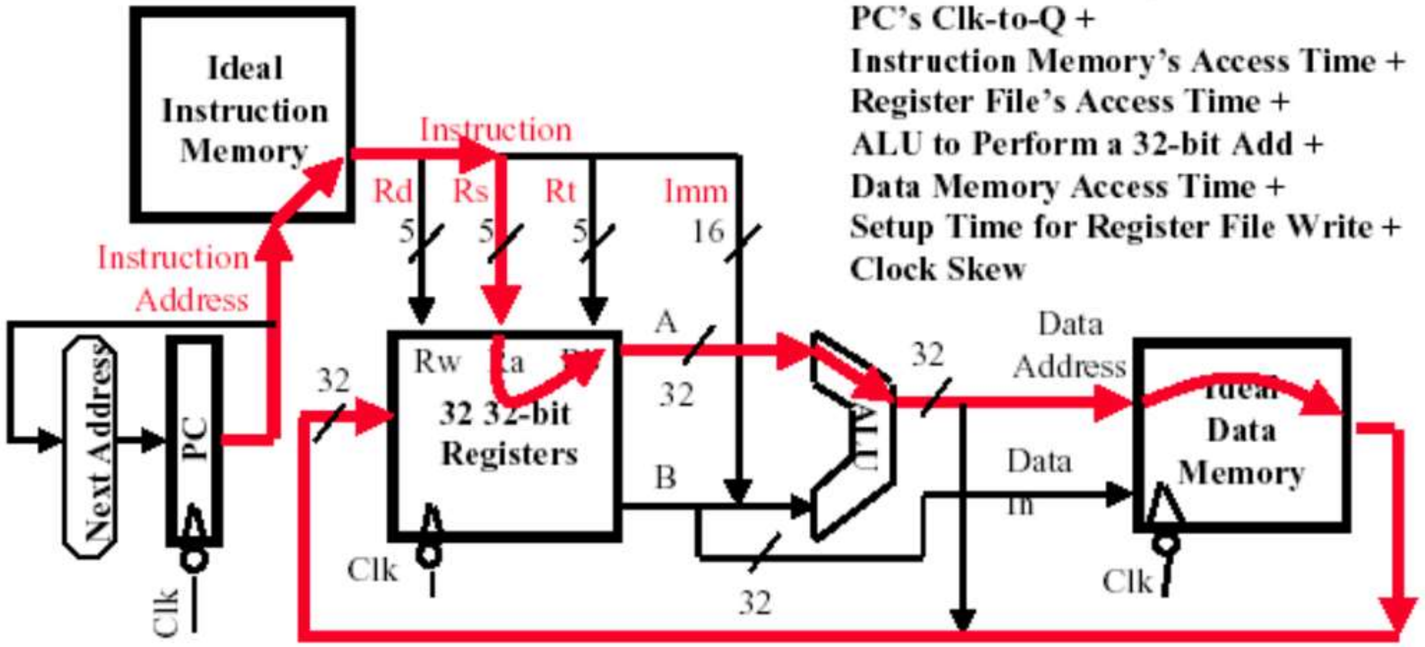


Control Inputs and Paths for BRZ



Abstract View of Critical Path

- Register file and ideal memory:
  - The CLK input is a factor ONLY during write operation
  - During read operation, behave as combinational logic:
    - Address valid => Output valid after "access time."



Critical Path (Load Operation) =
PC's Clk-to-Q +
Instruction Memory's Access Time +
Register File's Access Time +
ALU to Perform a 32-bit Add +
Data Memory Access Time +
Setup Time for Register File Write +
Clock Skew