Very Large Scale Integration II - VLSI II

RF & Datapath & Single Cycle Control

Berna Örs Yalçın

ITU VLSI Laboratories

Istanbul Technical University

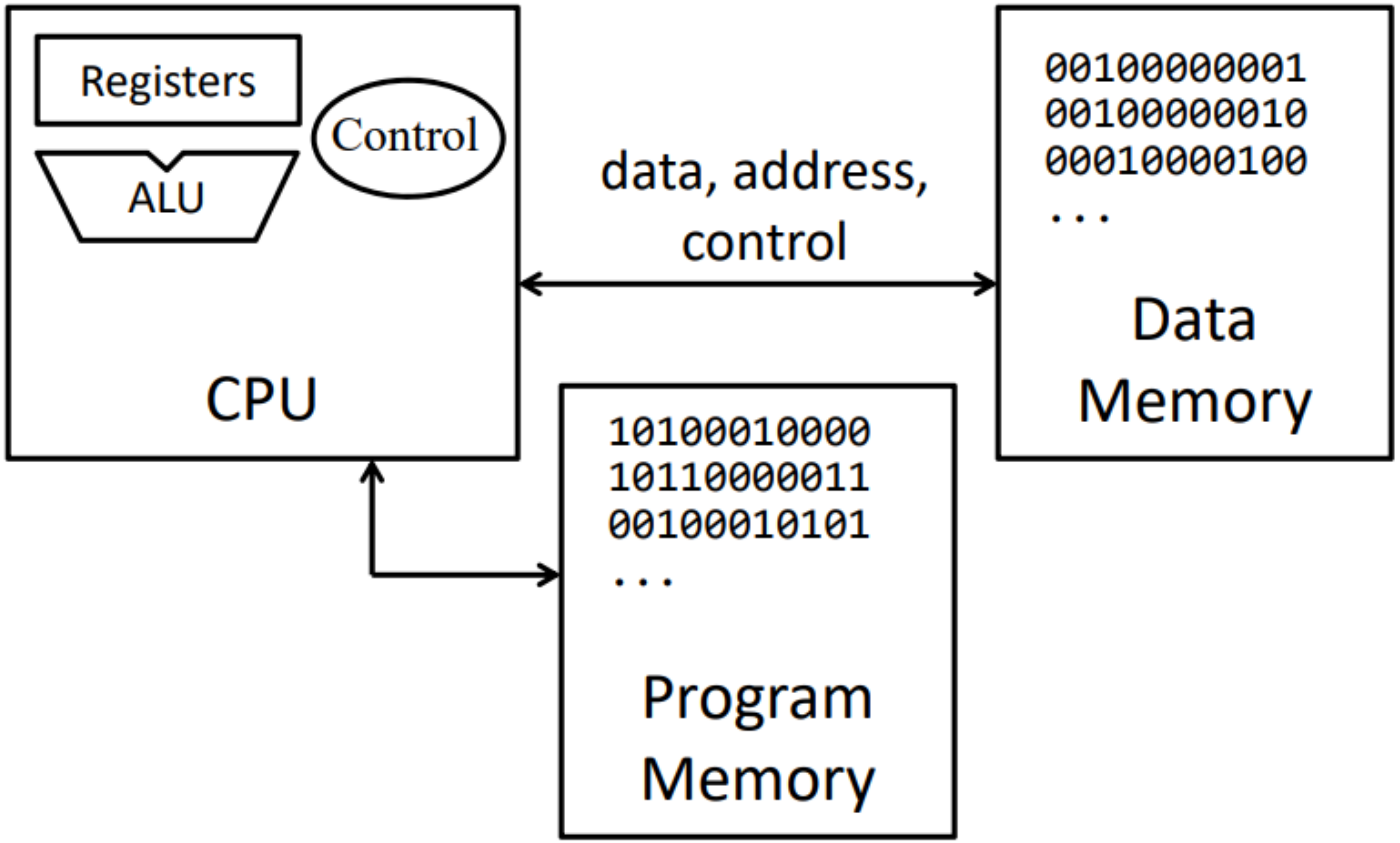
Topics for today

- Register File
- Generic Datapath
- Single Cycle Control and Issues

Basic Processor Structure

- The specification for a computer consists of a description of its appearance to a programmer at the lowest level, its *instruction set architecture* (*ISA*)
- From the ISA, a high-level description of the hardware to implement the computer, called the *computer architecture,* is formulated.
- This architecture, for a simple computer, is typically divided into two parts;
  - Datapath
  - Control Unit

Basic Processor Structure

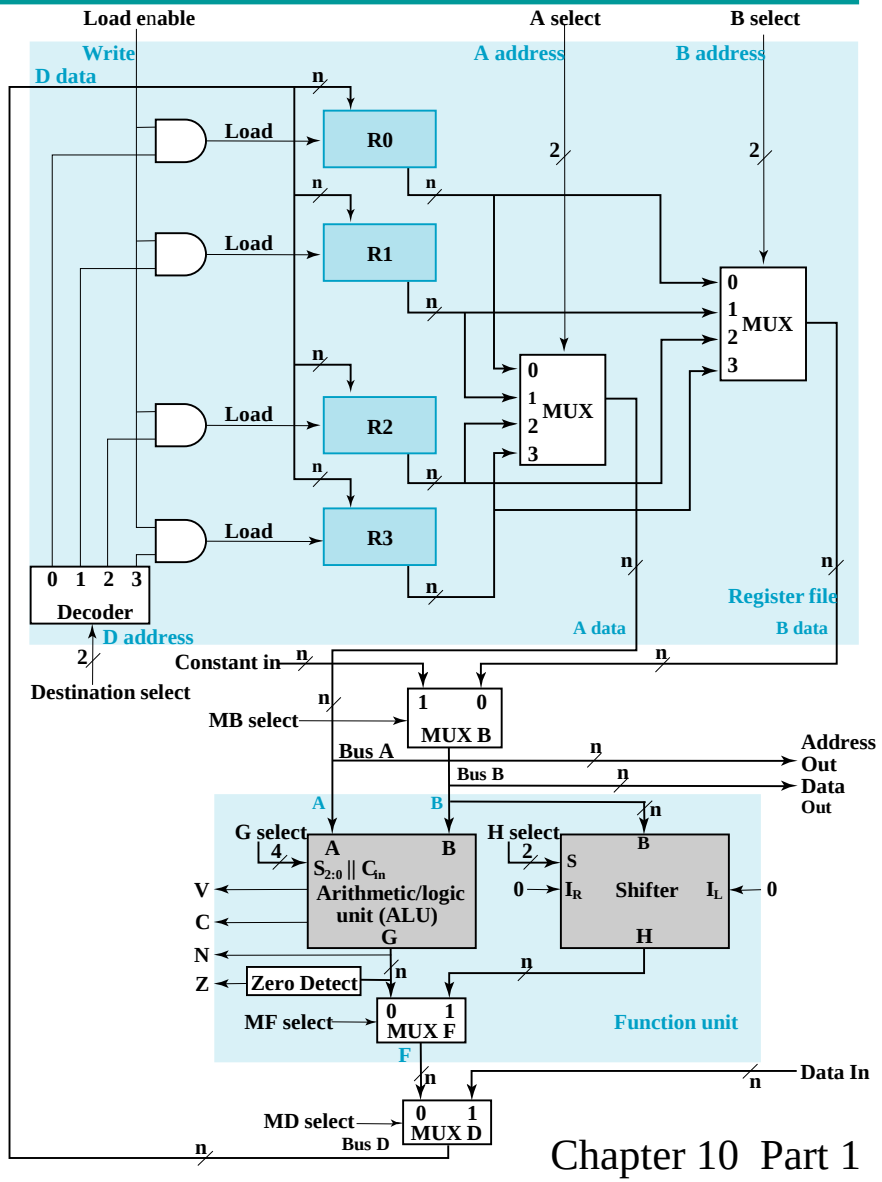


Basic Processor Structure

- The datapath is defined by a few basic components:
  - a set of registers (register file)
  - the microoperations performed on data stored in the registers (ALU)
  - the control interface
  - buses, multiplexers, decoders
- The control unit
  - provides signals that control the microoperations performed in the datapath and in other components of the system, such as memories.
  - controls its own operation, determining the sequence of events that occur. This sequence may depend upon the results of current and past microoperations executed.
  - directs the information flow through the buses, the ALU, the shifter, and the registers by applying signals to the select inputs.

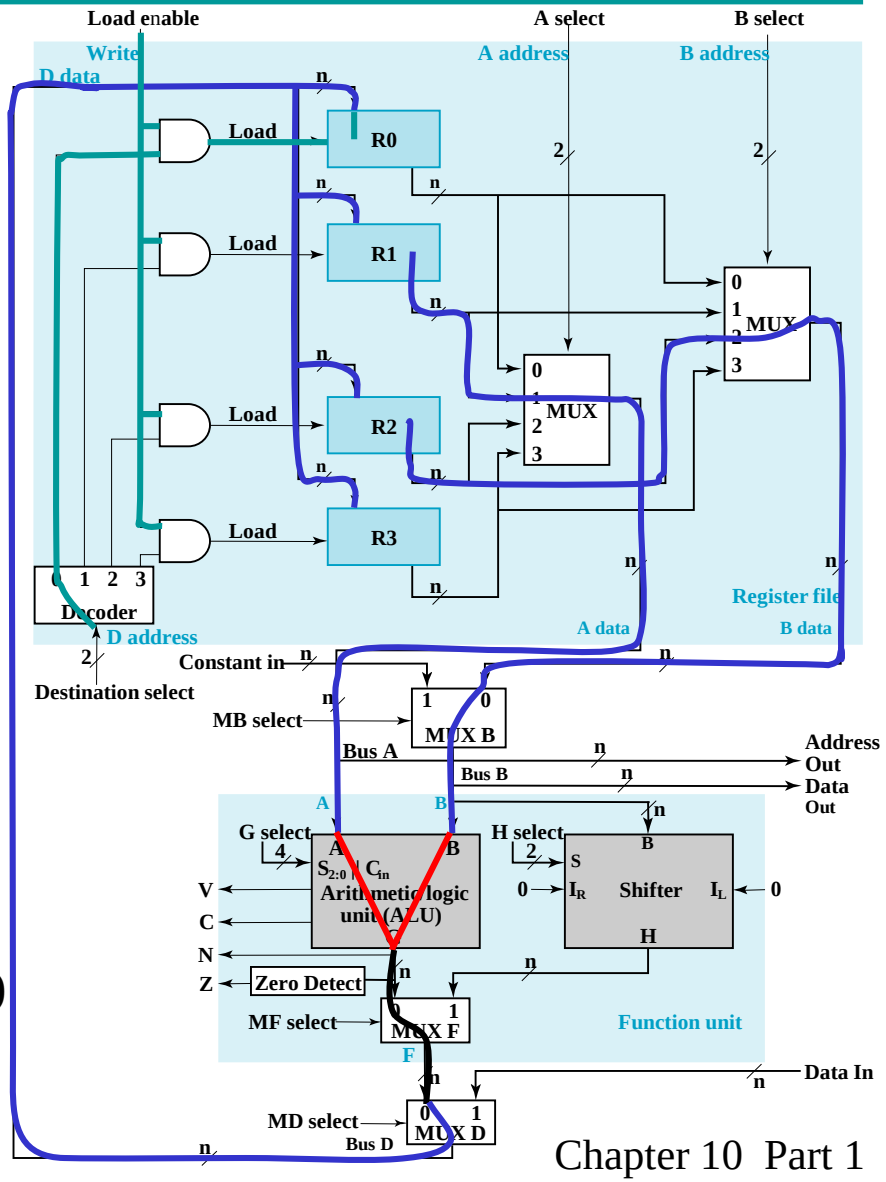
Datapath Example

- Four parallel-load registers
- Two mux-based register selectors
- Register destination decoder
- Mux B for external constant input
- Buses A and B with external address and data outputs
- ALU and Shifter with Mux F for output select
- Mux D for external data input
- Logic for generating status bits V, C, N, Z



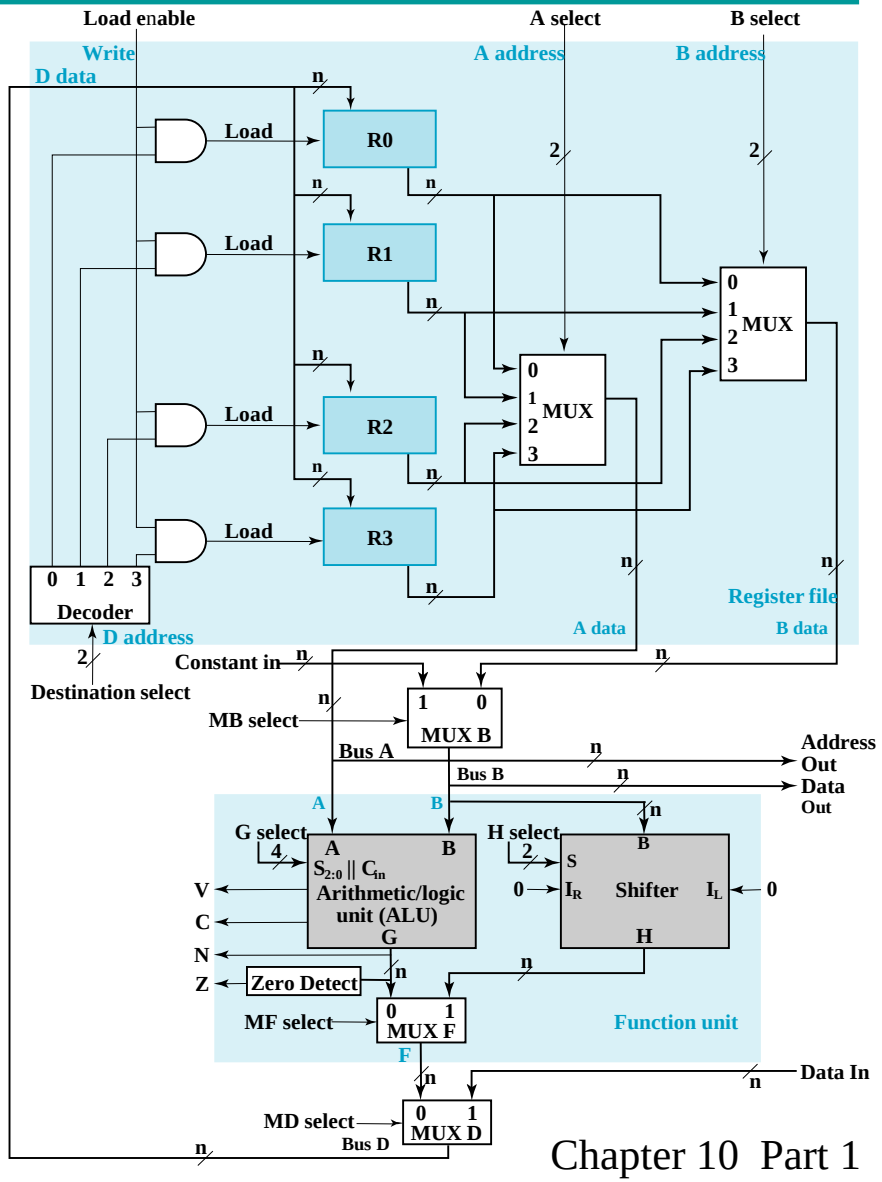
Datapath Example: Performing a Microoperation

- Microoperation: R0 ← R1 + R2
- Apply 01 to A select to place contents of R1 onto Bus A
- Apply 10 to B select to place contents of R2 onto B data and apply 0 to MB select to place B data on Bus B
- Apply 0010 to G select to perform addition G = Bus A + Bus B
- Apply 0 to MF select and 0 to MD select to place the value of G onto BUS D
- Apply 00 to Destination select to enable the Load input to R0
- Apply 1 to Load Enable to force the Load input to R0 to 1 so that R0 is loaded on the clock pulse (not shown)
- The overall microoperation requires 1 clock cycle



Datapath Example: Key Control Actions for Microoperation Alternatives

- Perform a shift microoperation – apply 1 to MF select
- Use a constant in a micro-operation using Bus B – apply 1 to MB select
- Provide an address and data for a memory or output write microoperation – apply 0 to Load enable to prevent register loading
- Provide an address and obtain data for a memory or output read microoperation – apply 1 to MD select
- For some of the above, other control signals become don't cares



Register File

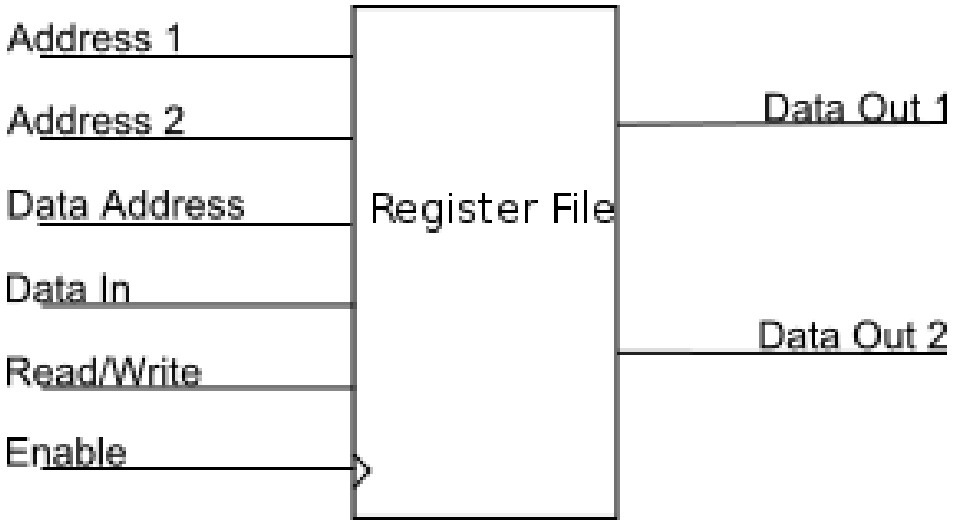
- Register File is
  - a block of processor registers.
  - defined in ISA. It is the first block used to store data hierarchically.
  - a special type of fast memory that permits one or more words to be read and one or more words to be written, all simultaneously.
  - visible to the programmers.
- The size of the register file is $2^m \times n$, where m is the number of register address bits and n is the number of bits per register.

Register File

- ISA defines both special purpose registers and general purpose registers.

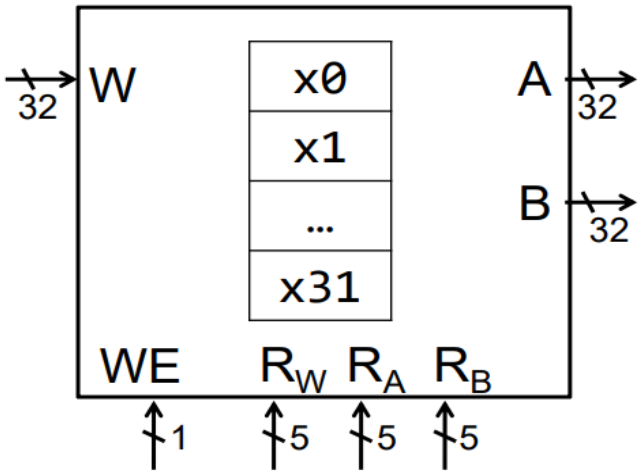
- Compiler makes arrangements according to the Register File which is defined in ISA.
  - Zero register
  - Stack pointer register
  - Return address register
  - etc

Register File Structure



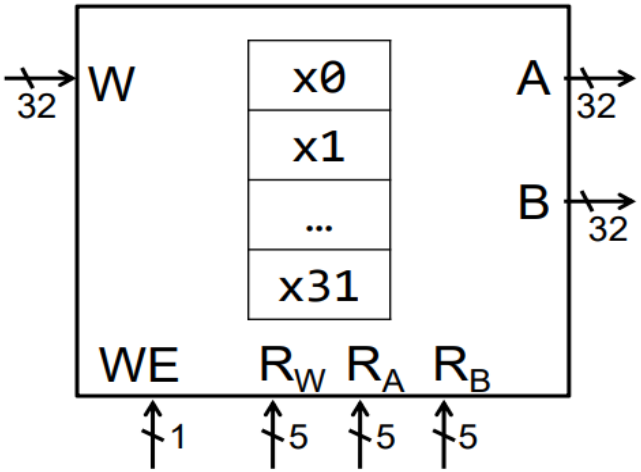
Register File for RISC-V ISA

- **WE:** Write Enable
- **W:** Data input for write operation
- **RW:** Write register address
- **RA-RB:** Read register address
- **A-B:** Data outputs from the readed registers



Register File for RISC-V ISA

- 32-bit fixed 32 registers numbered from 0 to 31
- Referred by numbers x0, x1, x2 ... x31
  - All have conventional names:
  - Example: x10 – x17 => a0 – a7, x28 – x31 => t3 – t6

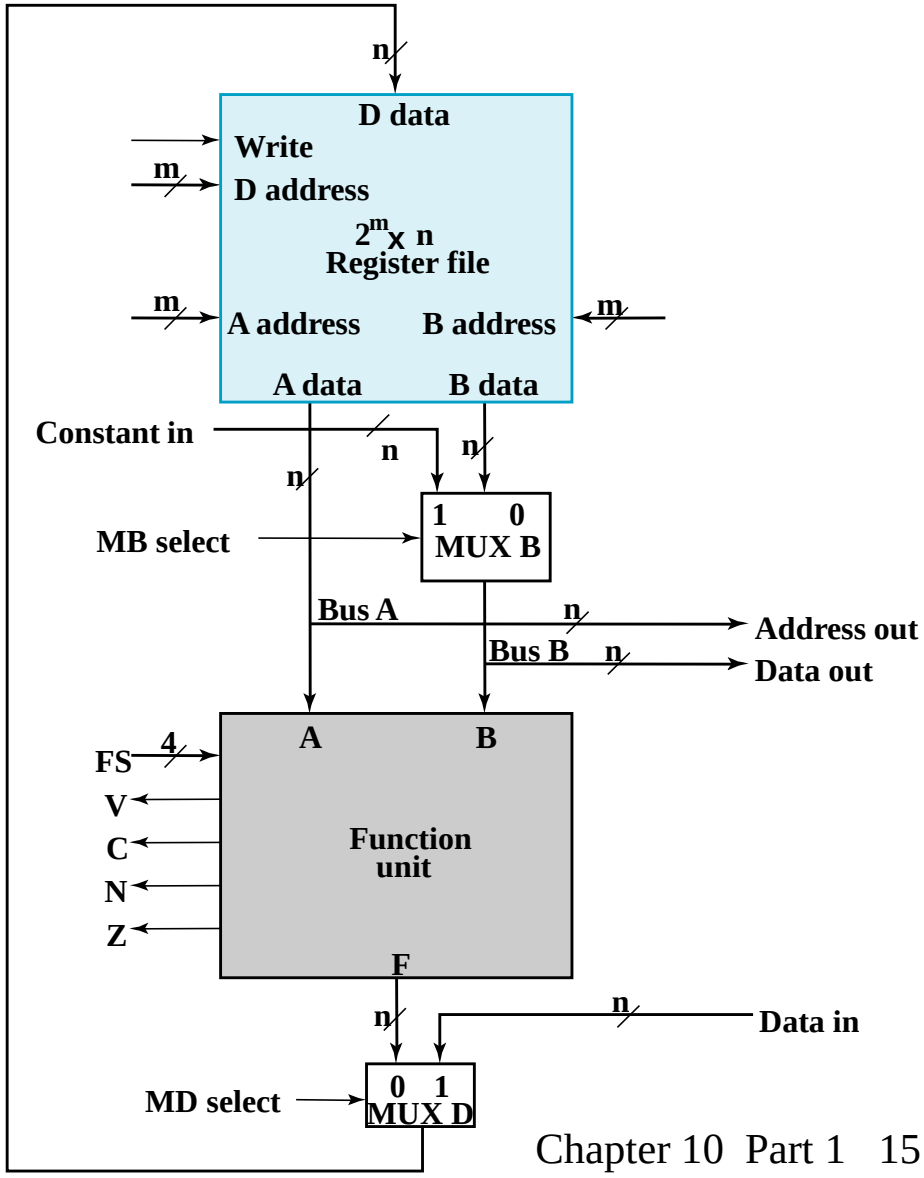


Register File for RISC-V ISA

Register	ABI Name	Description	Saver
x0	zero	hardwired zero	-
x1	ra	return address	Caller
x2	sp	stack pointer	Callee
x3	gp	global pointer	-
x4	tp	thread pointer	-
x5-7	t0-2	temporary registers	Caller
x8	s0 / fp	saved register / frame pointer	Callee
x9	s1	saved register	Callee
x10-11	a0-1	function arguments / return values	Caller
x12-17	a2-7	function arguments	Caller
x18-27	s2-11	saved registers	Callee
x28-31	t3-6	temporary registers	Caller

Datapath Representation

- The registers, and the multiplexer, decoder, and enable hardware for accessing them become a *register file*
- The ALU, shifter, MUX F and status hardware become a *function unit*
- In the register file:
  - Multiplexer select inputs become A address and B address
  - Decoder input becomes D address
  - Multiplexer outputs become A data and B data
  - Input data to the registers becomes D data
  - Load enable becomes write
- The G select, H select and MF select become FS



Definition of Function Unit Select (FS) Codes

FS(3:0)	MF Select	G Select(3:0)	H Select(3:0)	Microoperation
0000	0	0000	XX	$F \leftarrow A$
0001	0	0001	XX	$F \leftarrow A + 1$
0010	0	0010	XX	$F \leftarrow A + B$
0011	0	0011	XX	$F \leftarrow A + B + 1$
0100	0	0100	XX	$F \leftarrow A + \overline{B}$
0101	0	0101	XX	$F \leftarrow A + \overline{B} + 1$
0110	0	0110	XX	$F \leftarrow A - 1$
0111	0	0111	XX	$F \leftarrow A$
1000	0	1X00	XX	$F \leftarrow A \wedge B$
1001	0	1X01	XX	$F \leftarrow A \vee B$
1010	0	1X10	XX	$F \leftarrow A \oplus B$
1011	0	1X11	XX	$F \leftarrow \overline{A}$
1100	1	XXXX	00	$F \leftarrow B$
1101	1	XXXX	01	$F \leftarrow \text{sr } B$
1110	1	XXXX	10	$F \leftarrow \text{sl } B$

Boolean Equations:

MFS = F_3 F_2

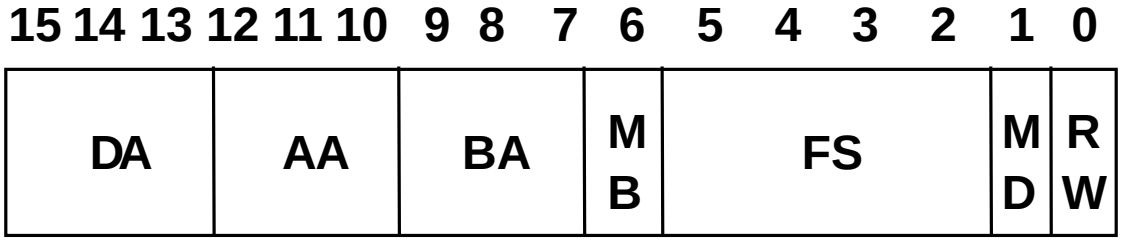
$GS_i = F_i$

$HS_i = F_i$

Control Word

- The datapath has many control inputs
- The signals driving these inputs can be defined and organized into a *control word*
- To execute a microinstruction, we apply control word values for a clock cycle. For most microoperations, the positive edge of the clock cycle is needed to perform the register load
- The datapath control word format and the field definitions are shown on the next slide

Control Word

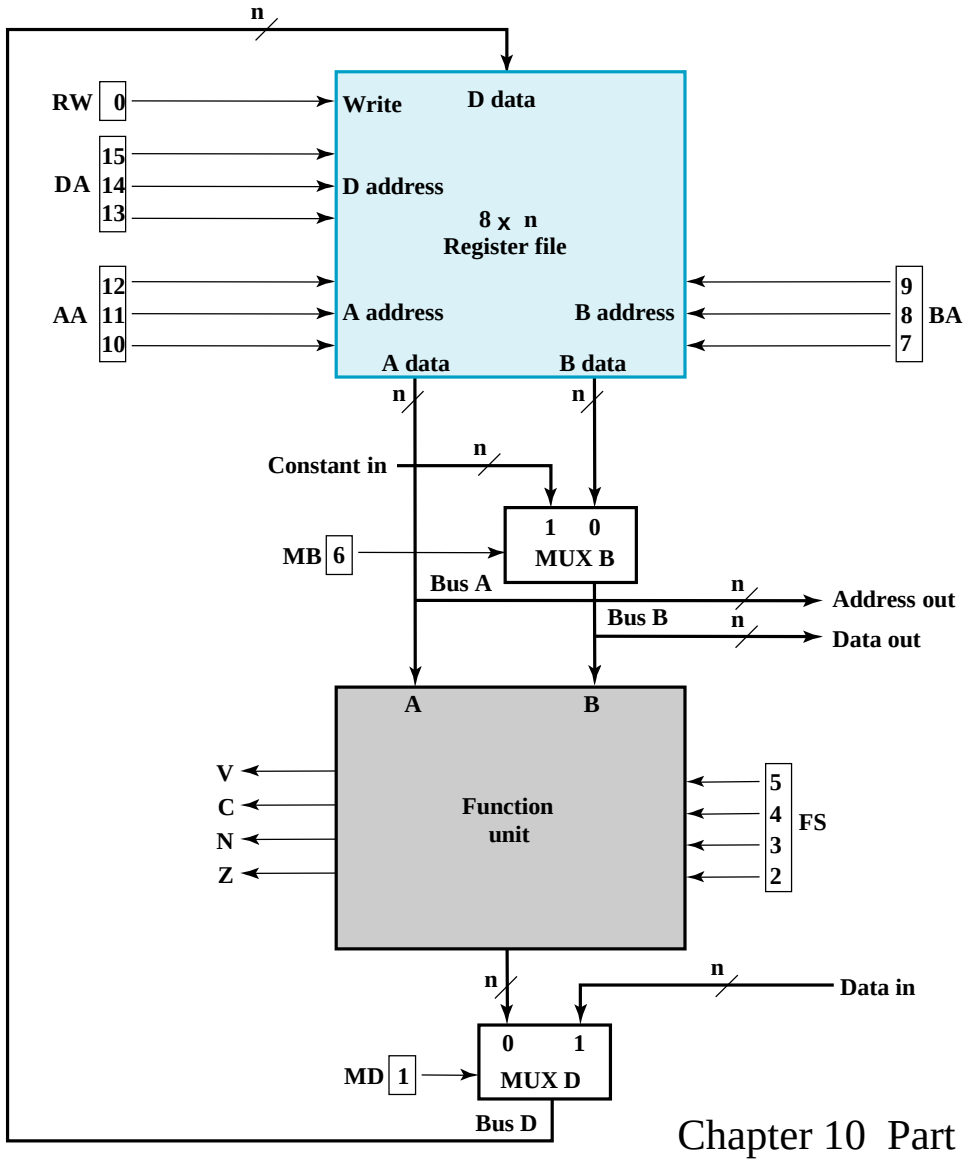


Control word

- DA – D Address
- AA – A Address
- BA – B Address
- MB – Mux B
- FS – Function Select
- MD – Mux D
- RW – Register Write

Control Word Block Diagram

- DA – D Address
- AA – A Address
- BA – B Address
- MB – Mux B
- FS – Function Select
- MD – Mux D
- RW – Register Write



Control Word Encoding

DA, AA, BA		MB		FS		MD		RW	
Function	**Code**	**Function**	**Code**	**Function**	**Code**	**Function**	**Code**	**Function**	**Code**
$R0$	000	Register	0	$F \leftarrow A$	0000	Function	0	No write	0
$R1$	001	Constant	1	$F \leftarrow A + 1$	0001	Data In	1	Write	1
$R2$	010			$F \leftarrow A + B$	0010				
$R3$	011			$F \leftarrow A + \overline{B} + 1$	0011				
$R4$	100			$F \leftarrow A + \overline{B}$	0100				
$R5$	101			$F \leftarrow A + \overline{B} + 1$	0101				
$R6$	110			$F \leftarrow A - 1$	0110				
$R7$	111			$F \leftarrow A$	0111				
				$F \leftarrow A \wedge B$	1000				
				$F \leftarrow A \vee B$	1001				
				$F \leftarrow A \oplus B$	1010				
				$F \leftarrow \overline{A}$	1011				
				$F \leftarrow B$	1100				
				$F \leftarrow \text{sr } B$	1101				
				$F \leftarrow \text{sl } B$	1110				

Microoperations for the Datapath - Symbolic Representation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	$R1$	$R2$	$R3$	Register	$F = A + \overline{B} + 1$	Function	Write
$R4 \leftarrow$ sl R6	$R4$	—	$R6$	Register	$F =$ sl B	Function	Write
$R7 \leftarrow R7 + 1$	$R7$	$R7$	—	Register	$F = A + 1$	Function	Write
$R1 \leftarrow R0 + 2$	$R1$	$R0$	—	Constant	$F = A + B$	Function	Write
Data out $\leftarrow R3$	——		$R3$	Register	—	—	No Write
$R4 \leftarrow$ Data in	$R4$	——		—	—	Data in	Write
$R5 \leftarrow 0$	$R5$	$R0$	$R0$	Register	$F = A \oplus B$	Function	Write

Microoperations for the Datapath - Binary Representation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	0101	0	1
$R4 \leftarrow$ sl R6	100	XXX	110	0	1110	0	1
$R7 \leftarrow R7 + 1$	111	111	XXX	0	0001	0	1
$R1 \leftarrow R0 + 2$	001	000	XXX	1	0010	0	1
Data out $\leftarrow R3$	XXX	XXX	011	0	XXXX	X	0
$R4 \leftarrow$ Data in	100	XXX	XXX	X	XXXX	1	1
$R5 \leftarrow 0$	101	000	000	0	1010	0	1

The Entire Datapath

- Putting it all together (datapath):
  - Arithmetic Logic Unit (ALU)
  - Register File
  - Memory
    - SRAM
    - DRAM

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

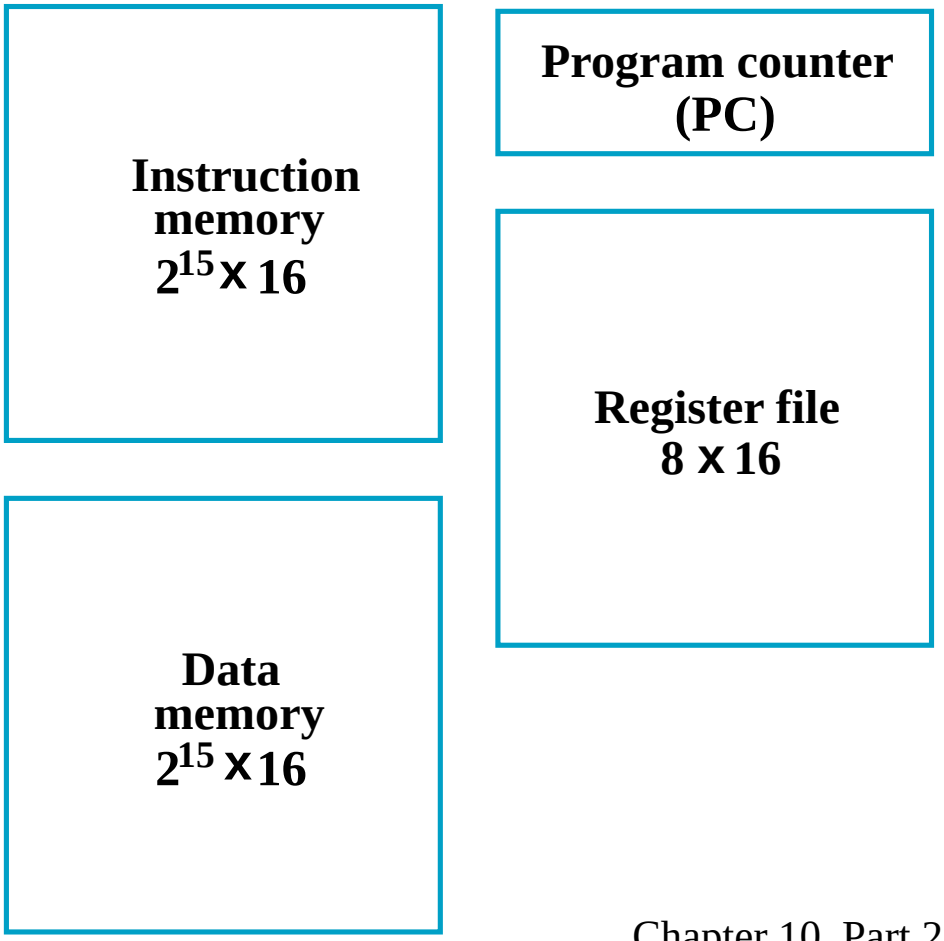
- A programmable system uses a sequence of *instructions* to control its operation
- An typical instruction specifies:
  - Operation to be performed
  - Operands to use, and
  - Where to place the result, or
  - Which instruction to execute next
- Instructions are stored in RAM or ROM as a *program*
- The addresses for instructions in a computer are provided by a *program counter (PC)* that can
  - Count up
  - Load a new address based on an instruction and, optionally, status information

Instruction Set Architecture (ISA) (continued)

- **The PC and associated control logic are part of the Control Unit**
- **Executing an instruction - activating the necessary sequence of operations specified by the instruction**
- **Execution is controlled by the control unit and performed:**
  - **In the datapath**
  - **In the control unit**
  - **In external hardware such as memory or input/output**

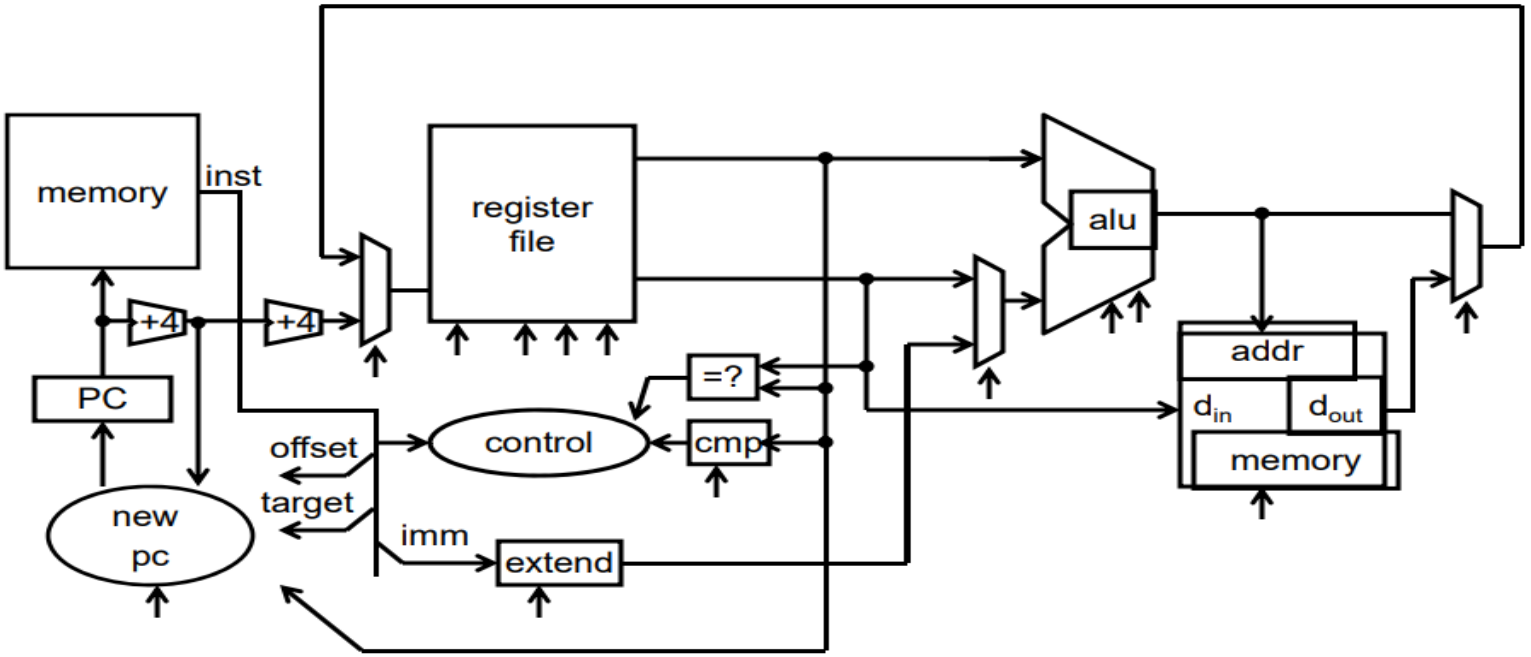
ISA: Storage Resources

- The storage resources are "visible" to the programmer at the lowest software level (typically, machine or assembly language)
- **Storage resources for the SC =>**
- Separate instruction and data memories imply "Harvard architecture"
- Done to permit use of single clock cycle per instruction implementation
- Due to use of "cache" in modern computer architectures, is a fairly realistic model

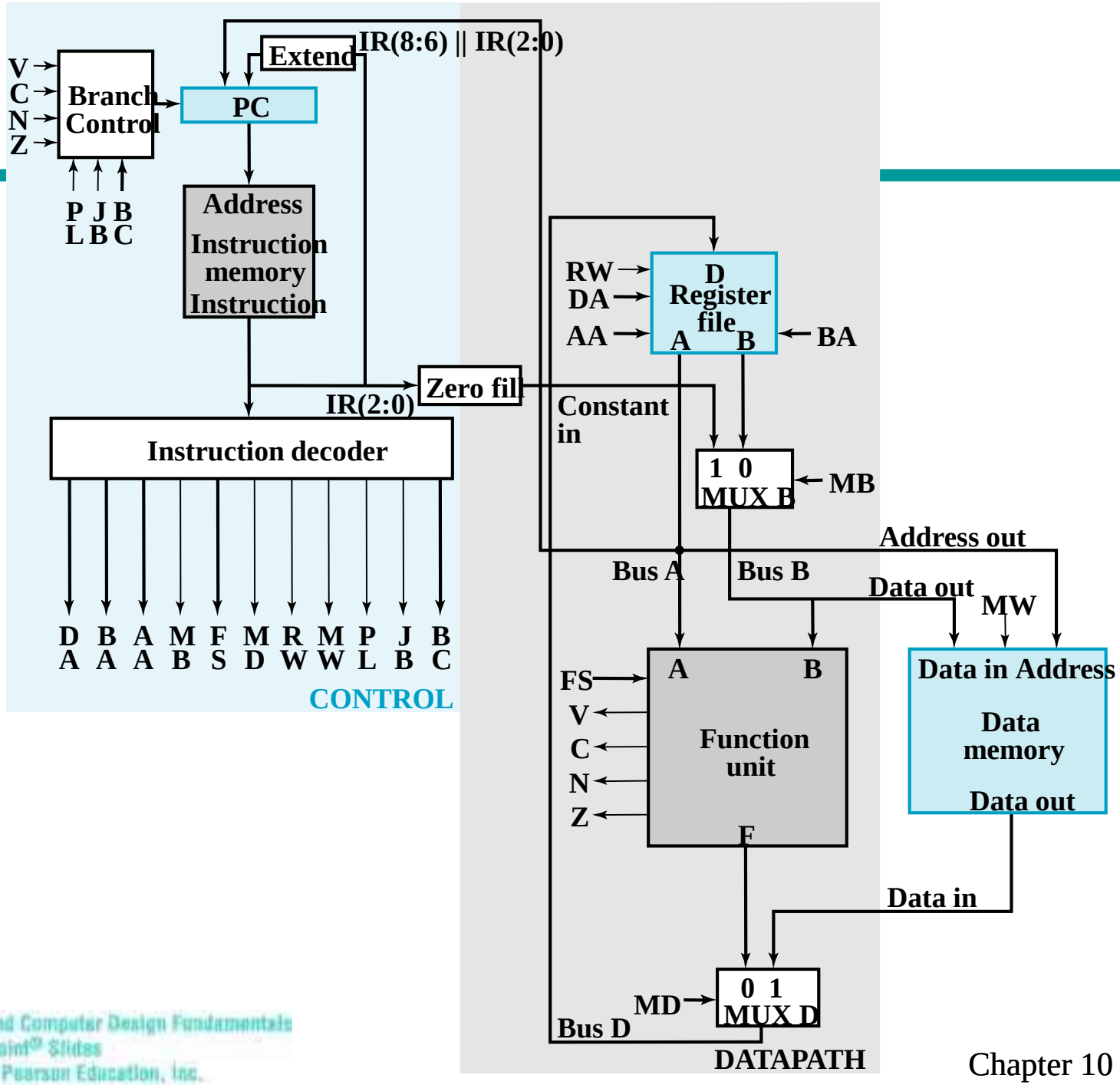


Single Cycle Processor

Big Picture: Building a Processor



A single cycle processor



The Control Unit

- The Data Memory has been attached to the Address Out and Data Out and Data In lines of the Datapath.
- The MW input to the Data Memory is the Memory Write signal from the Control Unit.
- For convenience, the Instruction Memory, which is not usually a part of the Control Unit is shown within it.
- 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

PC Function

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

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

- In addition to the above register transfers, the PC must also implement: PC ← PC + 1
- The first two transfers above 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]
- The counting function of the PC requires addition to the PC of 1

PC Function (continued)

- **Branch Control determines the PC transfers based on five of its inputs defined as follows:**
  - **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.**
- **The above is summarize by the following table:**

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

- **Sufficient information is provided here to design the PC**

Instruction Decoder

- **The combinational instruction decoder converts the instruction into the signals necessary to control all parts of the computer during the single cycle execution**
- **The input is the 16-bit Instruction**
- **The outputs are control signals:**
  - **Register file addresses DA, AA, and BA,**
  - **Function Unit Select FS**
  - **Multiplexer Select Controls MB and MD,**
  - **Register file and Data Memory Write Controls RW and MW, and**
  - **PC Controls PL, JB, and BC**
- **The register file outputs are simply pass-through signals:**
  DA = DR, AA = SA, and BA = SB

Determination of the remaining signals is more complex.

Instruction Decoder (continued)

- **The remaining control signals do not depend on the addresses, so must be a function of IR(13:9)**
- **Formulation requires examining relationships between the outputs and the opcodes given before.**
- **Observe that for other than branches and jumps, FS = IR(12:9)**
- **This implies that the other control signals should depend as much as possible on IR(15:13) (which actually were assigned with decoding in mind!)**
- **To make some sense of this, we divide instructions into types as shown in the table on the next page**

Instruction Decoder (continued)

Truth Table for Instruction Decoder Logic

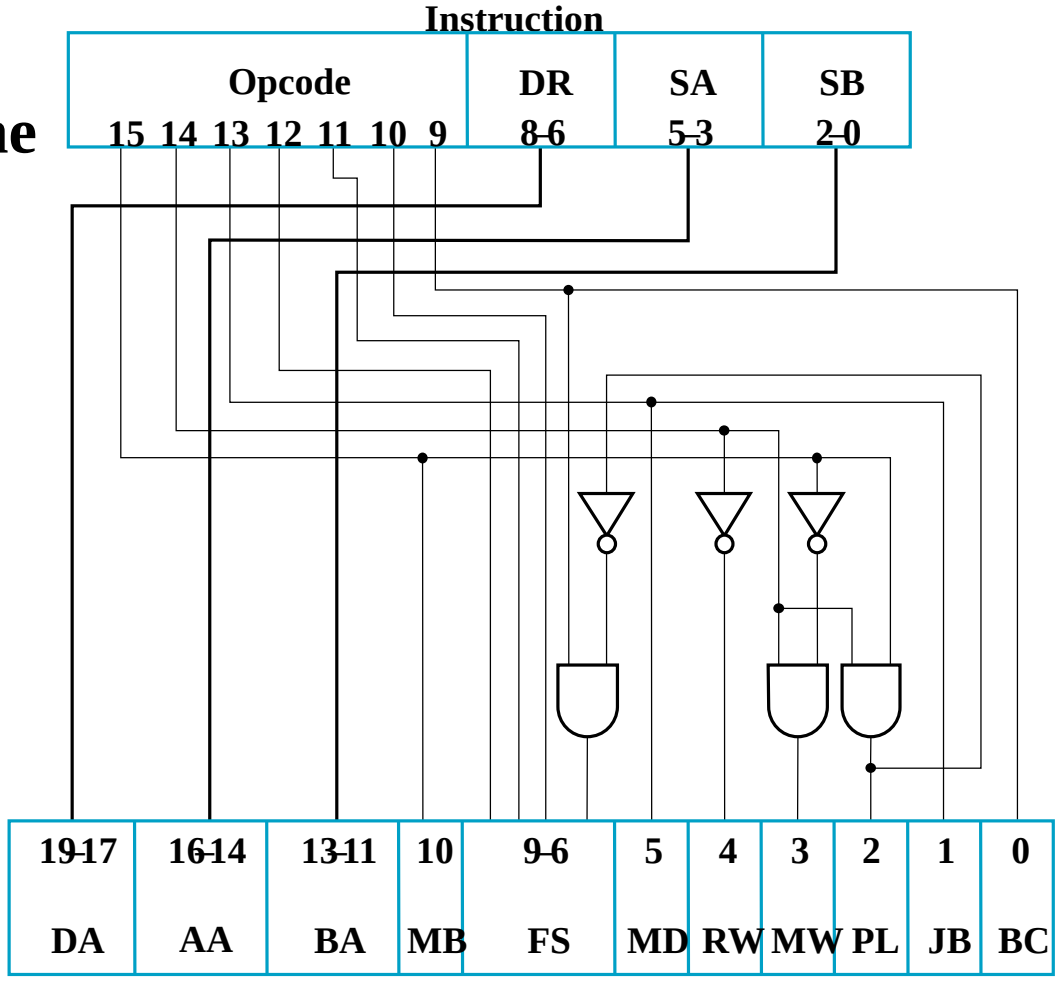
	Instruction Bits				Control Word Bits						
Instruction Function Type	**15**	**14**	**13**	**9**	**MB**	**MD**	**RW**	**MW**	**PL**	**JB**	**BC**
Function unit operations using registers	0	0	0	X	0	0	1	0	0	X	X
Memory read	0	0	1	X	0	1	1	0	0	X	X
Memory write	0	1	0	X	0	X	0	1	0	X	X
Function unit operations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

Instruction Decoder (continued)

- **The types are based on the blocks controlled and the seven signals to be generated; types can be divided into two groups:**
  - **Datapath and Memory Control (First 4 types)**
  - **PC Control (Last 3 types)**
- **In Datapath and Memory Control blocks controlled are considered:**
  - **Mux B (1st and 4th types)**
  - **Memory and Mux D (2nd and 3rd types)**
  - **By assigning codes with no or only one 1 for these, implementation of MB, MD, RW and MW are simplified.**
- **In Control Unit more of a bit setting approach was used:**
  - **Bit 15 = Bit 14 = 1 were assigned to generate PL**
  - **Bit 13 values were assigned to generate JB.**
  - **Bit 9 was use as BC which contradicts FS = 0000 needed for branches. To force FS(6) to 0 for branches, Bit 9 into FS(6) is disabled by PL.**
- **Also, useful bit correlations between values in the two groups were exploited in assigning the codes.**

Instruction Decoder (continued)

- **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**



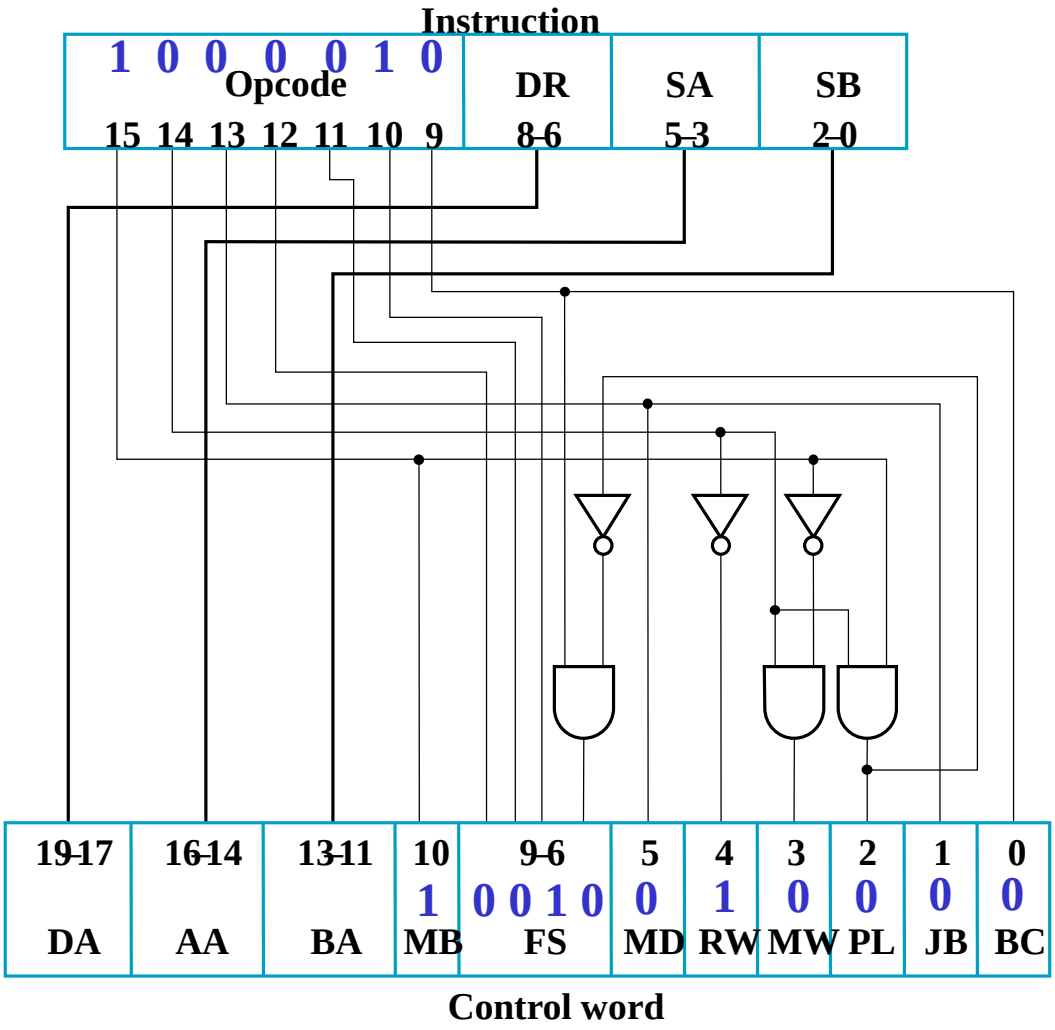
Example Instruction Execution

Six Instructions for the Single-Cycle Computer

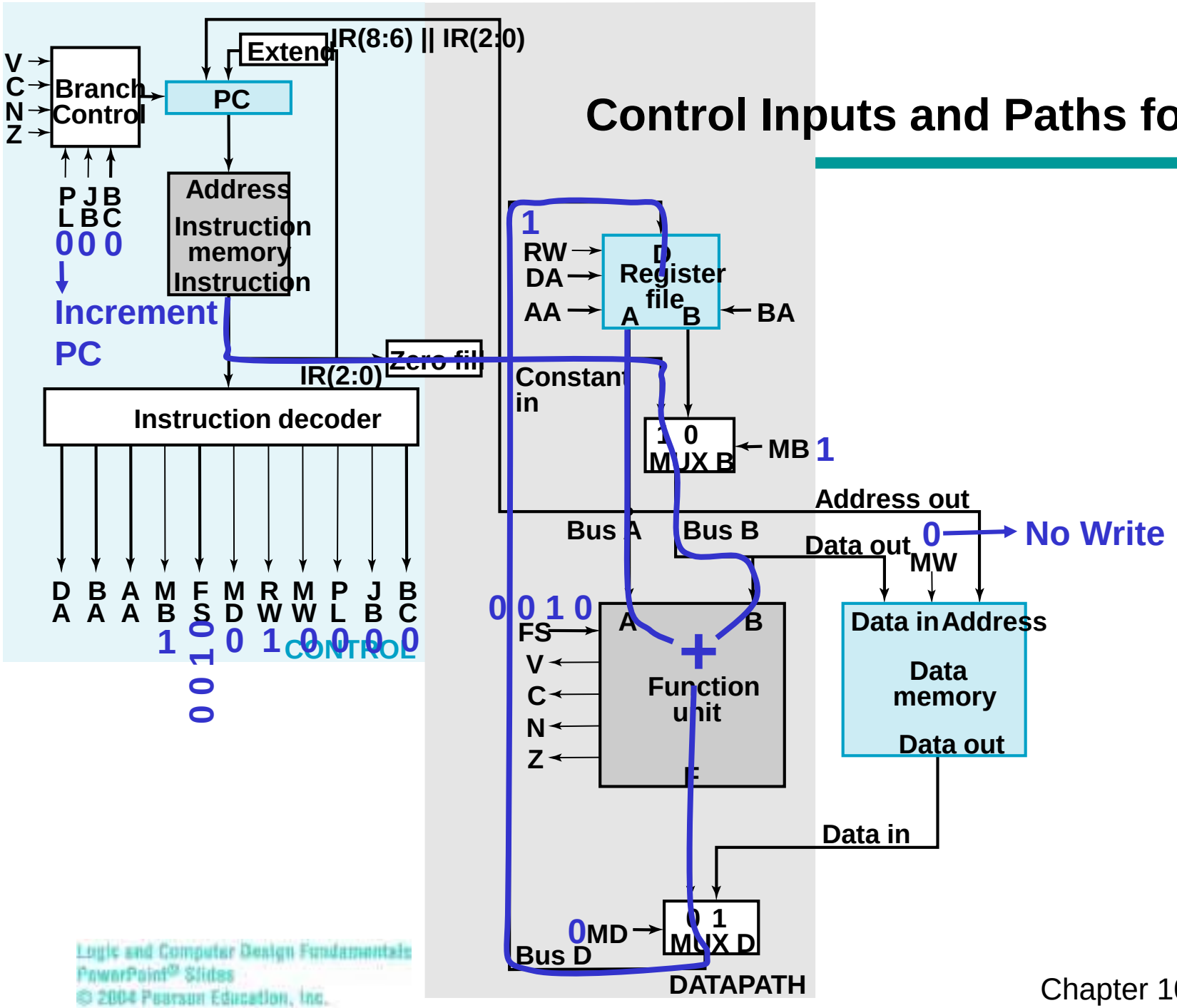
Operation code	Symbolic name	Format	Description	Function	MB	MD	RW	MW	PL	JB	BC
1000010	ADI	Immediate	Add immediate operand	$R[DR] \leftarrow R[SA] + zf\,I(2:0)$	1	0	1	0	0	0	0
0010000	LD	Register	Load memory content into register	$R[DR] \leftarrow M[R[SA]]$	0	1	1	0	0	1	0
0100000	ST	Register	Store register content in memory	$M[R[SA]] \leftarrow R[SB]$	0	1	0	1	0	0	0
0001110	SL	Register	Shift left	$R[DR] \leftarrow sl\,R[SB]$	0	0	1	0	0	1	0
0001011	NOT	Register	Complement register	$R[DR] \leftarrow \overline{R[SA]}$	0	0	1	0	0	0	1
1100000	BRZ	Jump/Branch	If $R[SA] = 0$, branch to PC + se AD	If R[SA] = 0, $PC \leftarrow PC + se\,AD$, If $R[SA] \neq 0$, $PC \leftarrow PC + 1$	1	0	0	0	1	0	0

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

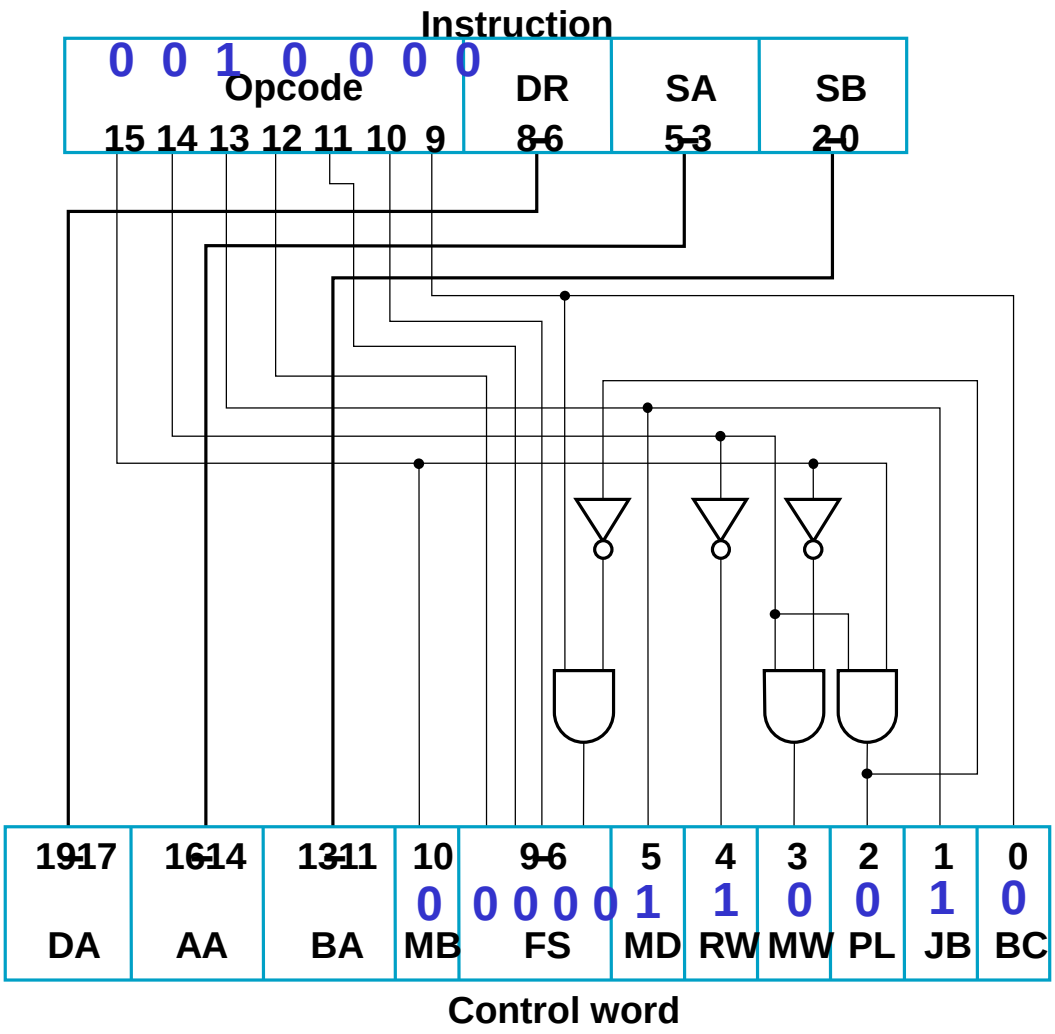
Decoding for ADI



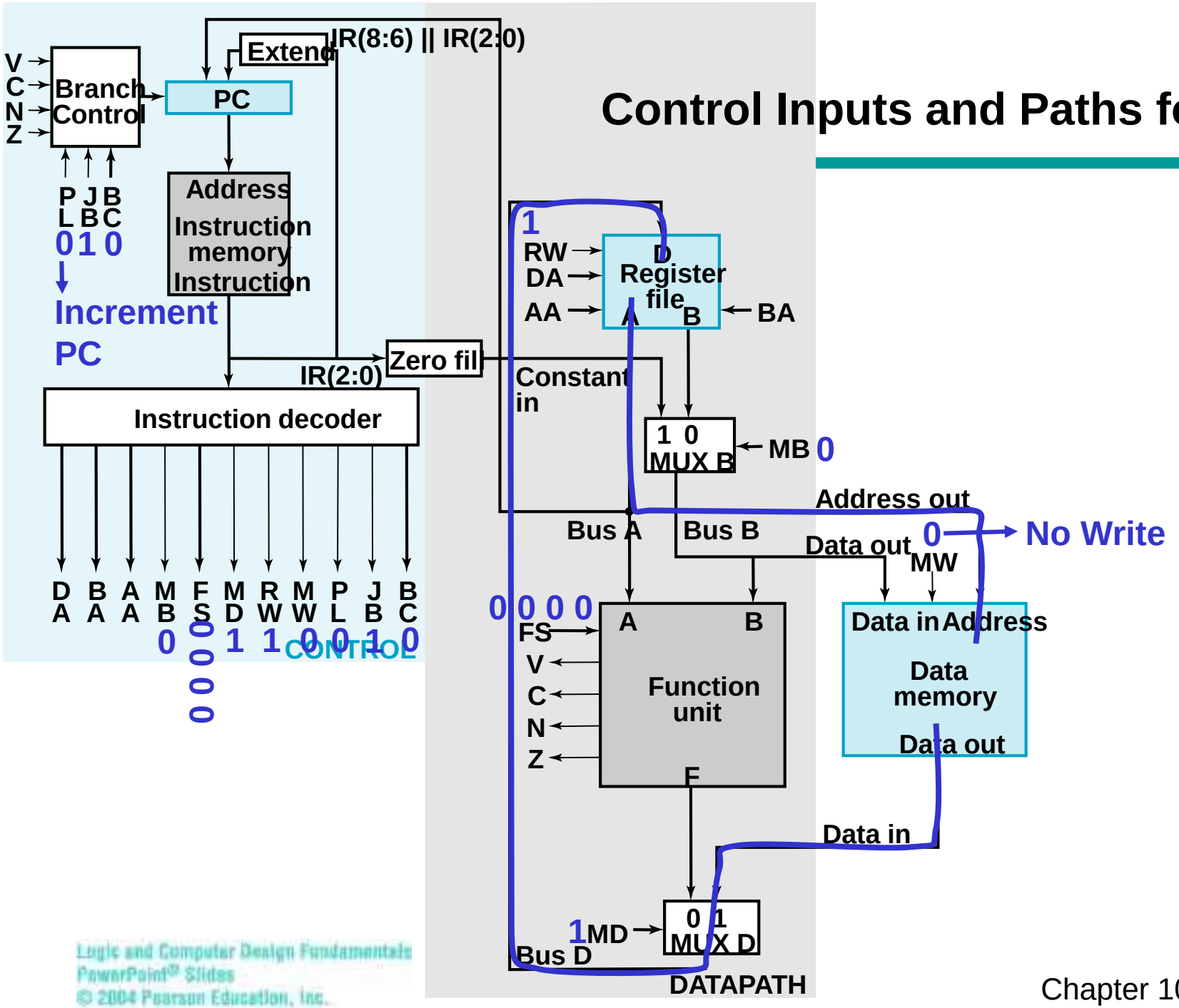
Control Inputs and Paths for ADI



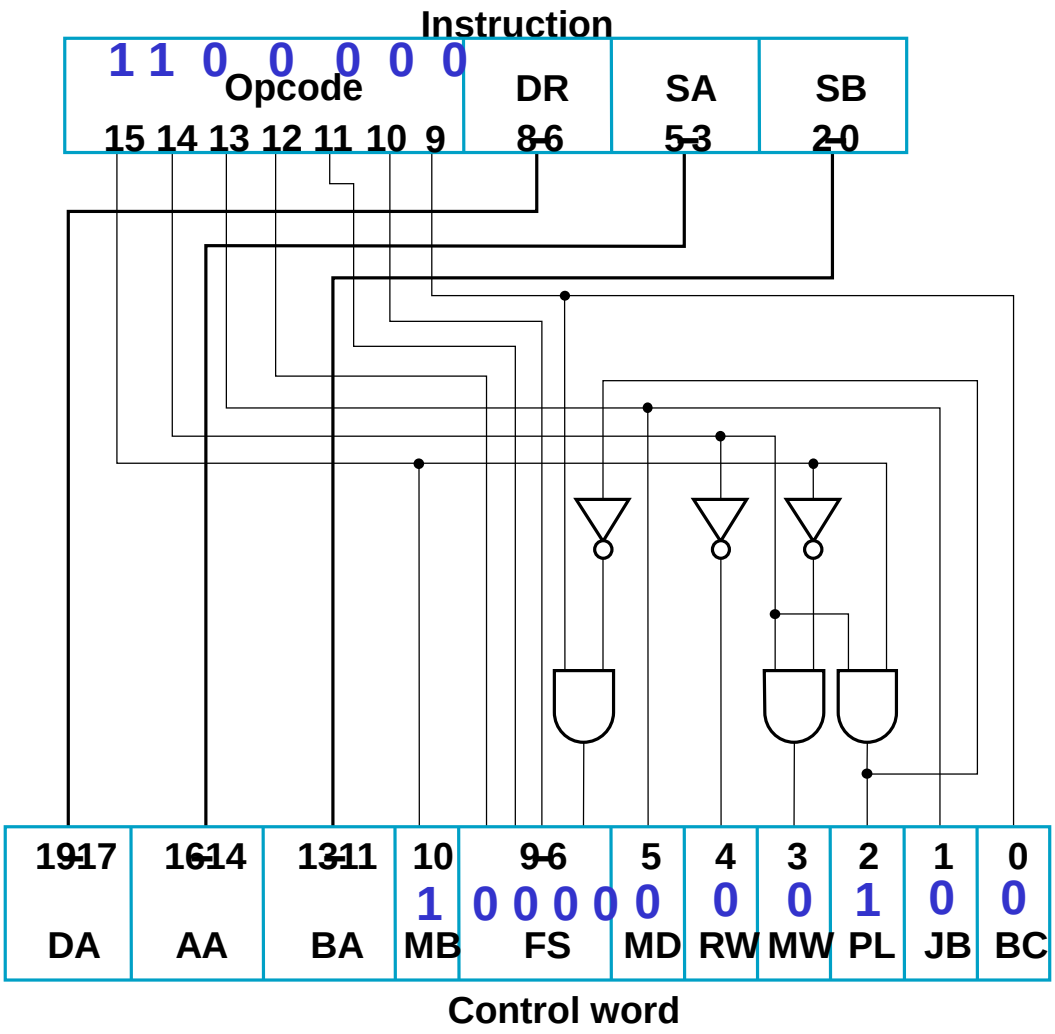
Decoding for LD



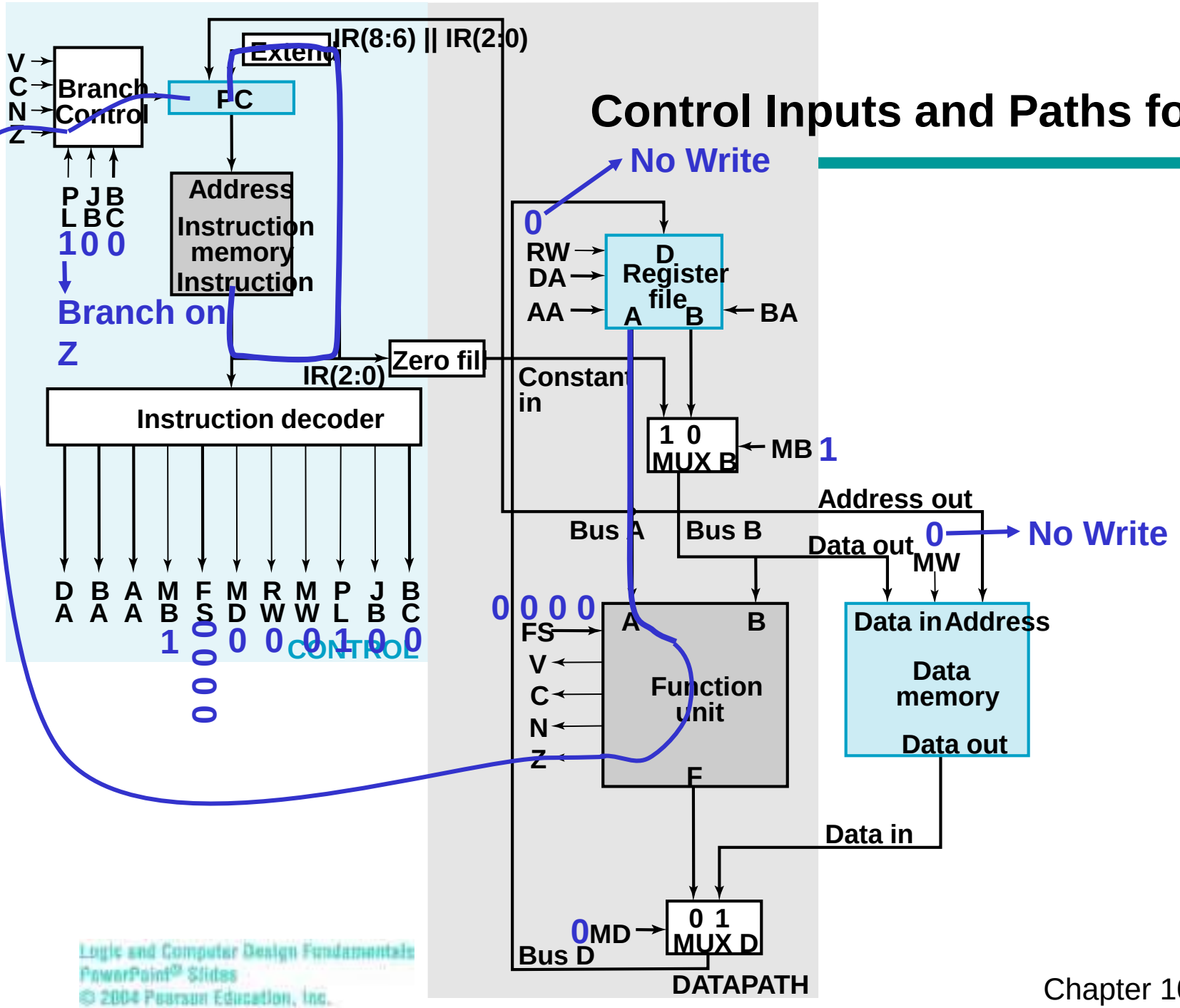
Control Inputs and Paths for LD



Decoding for BRZ



Control Inputs and Paths for BRZ



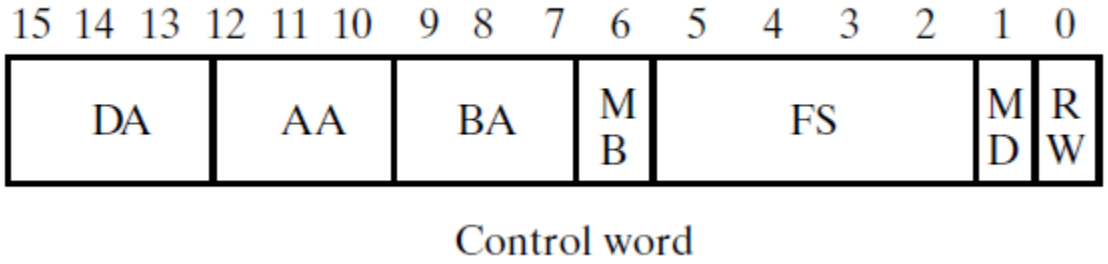
Single Cycle Control

- Imagine that you have designed all datapath.
- Then, you need apply a technique to control the datapath.
- One of the basic methods is called single cycle control. Basically, the all datapath operations are done in one clock cycle.

Single Cycle Control

- The selection variables for the datapath control the microoperations executed within the datapath for any given clock pulse.

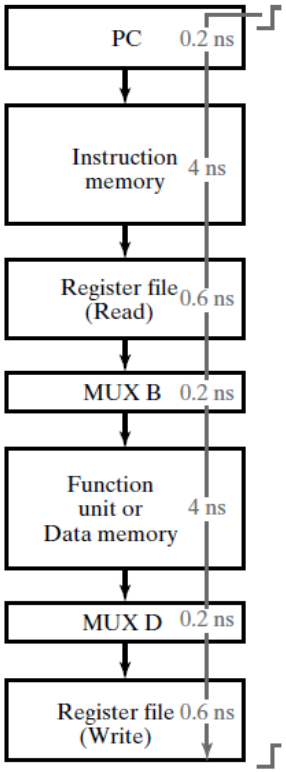
- The selection variables control the addresses for the data read from the register file, the function performed by the function unit, and the data loaded into the register file, as well as the selection of external data.



Control word

Single Cycle Control

- Delay path in a single cycle processor = delay of whole datapath



Single Cycle Control

- Although advantageous in terms of resource utilization and control circuit simplicity, there are some important issues with the single cycle control strategy.
- One is in the area of performing complex operations.
- For example, suppose that an instruction is desired that executes unsigned binary multiplication using a multiplication algorithm that processes one bit of the multiplier at a time.

Single Cycle Control

- With the given datapath, this cannot be accomplished by a microoperation that can be executed in a single clock cycle.
- Thus, a control organization that provides multiple clock cycles for the execution of instructions is needed.
- Finally, the single-cycle computer has a lower limit on the clock period based on a long worst-case delay path.

Single Cycle Control

- Also, the single-cycle computer has two distinct N-bit memories, one for instructions and one for data.

- For a simple computer with instructions and data in the same N-bit memory, two read accesses of memory are required to execute an instruction that loads a data word from memory into a register.

Single Cycle Control

- The first access obtains the instruction, and the second access, if required, reads or writes the data word.
- Since two different addresses must be applied to the memory address inputs, at least two clock cycles, one for each address, are required for obtaining and executing the instruction.
- All of these issues can be handled by using multi-cycle control strategy (subject of next week).

References

- Krste Asanovic, CS 152 Computer Architecture and Engineering, University of California at Berkeley, 2015.
- Logic and Computer Design Fundamentals,M. Morris Mano Charles Kime,Fourth Edition.
- Weatherspoon, Bala, Bracy, and Sirer, CS3410, Compter Science, Cornell University