Logic and Computer Design Fundamentals

Chapter 10 – Computer Design Basics

Part 2 – A Simple Computer

Charles Kime & Thomas Kaminski

© 2004 Pearson Education, Inc.
Terms of Use
(Hyperlinks are active in View Show mode)

Overview

- **Part 1 – Datapaths**
  - **Introduction**
  - **Datapath Example**
  - **Arithmetic Logic Unit (ALU)**
  - **Shifter**
  - **Datapath Representation and Control Word**
- **Part 2 – A Simple Computer**
  - **Instruction Set Architecture (ISA)**
  - **Single-Cycle Hardwired Control**
    - **PC Function**
    - **Instruction Decoder**
    - **Example Instruction Execution**
- **Part 3 – Multiple Cycle Hardwired Control**
  - **Single Cycle Computer Issues**
  - **Sequential Control Design**

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

© 2004 Pearson Education, Inc.

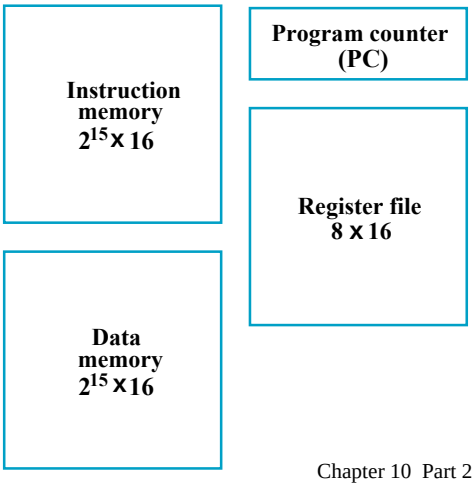
Instruction Set Architecture (ISA) (continued)

- 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

© 2004 Pearson Education, Inc.

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



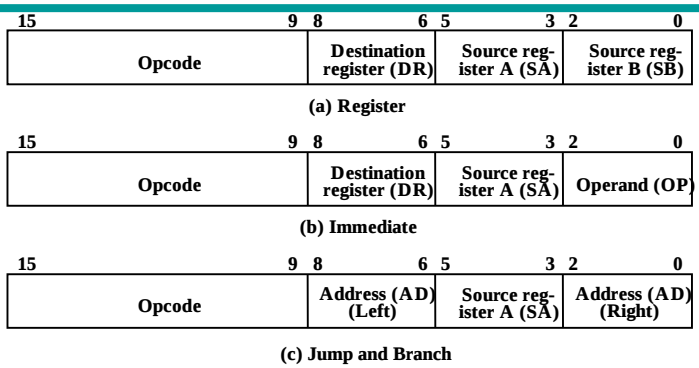
Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA: Instruction Format

- A instruction consists of a bit vector
- The *fields* of an instruction are subvectors representing specific functions and having specific binary codes defined
- The *format* of an instruction defines the subvectors and their function
- An ISA usually contains multiple formats
- The SC ISA contains the three formats presented on the next slide

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

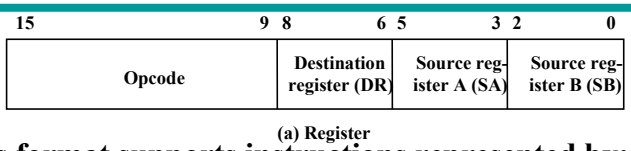
ISA: Instruction Format



- The three formats are: Register, Immediate, and Jump and Branch
- All formats contain an Opcode field in bits 9 through 15.
- The Opcode specifies the operation to be performed
- More details on each format are provided on the next three slides

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

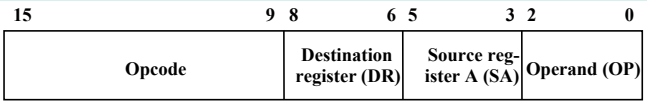
ISA: Instruction Format (continued)



- This format supports instructions represented by:
  - R1 ← R2 + R3
  - R1 ← sl R2
- There are three 3-bit register fields:
  - DR - specifies destination register (R1 in the examples)
  - SA - specifies the A source register (R2 in the first example)
  - SB - specifies 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?
  - The source for the shifter in our datapath to be used in implementation is Bus B rather than Bus A

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA: Instruction Format (continued)

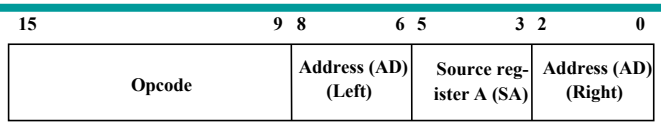


(b) Immediate

- This format supports instructions described by:
  - R1 ← R2 + 3
- The B Source Register field is replaced by an Operand field OP which specifies a constant.
- The Operand:
  - 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

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA: Instruction Format (continued)



(c) Jump and Branch

- This instruction supports changes in the sequence of instruction execution by adding an extended, 6-bit, signed 2s-complement *address offset* to the PC value
- The 6-bit Address (AD) field replaces the DR and SB fields
  - Example: Suppose that a jump is specified by the Opcode and the PC contains 45 (0…0101101) and Address contains – 12 (110100). Then the new PC value will be:
    0…0101101 + (1…110100) = 0…0100001 (45 + (– 12) = 33)
- The SA field is retained to permit jumps and branches on N or Z based on the contents of Source register A

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA: Instruction Specifications

- **The specifications provide:**
  - **The name of the instruction**
  - **The instruction's opcode**
  - **A shorthand name for the opcode called a *mnemonic***
  - **A specification for the instruction format**
  - **A register transfer description of the instruction, and**
  - **A listing of the status bits that are meaningful during an instruction's execution (not used in the architectures defined in this chapter)**

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA: Instruction Specifications (continued)

Instruction Specifications for the SimpleComputer - Part 1

Instruction	Opcode	Mnemonic	Format	Description	Status Bits
Move A	0000000	MOVA	RD,RA	R[DR] ← R[SA]	N, Z
Increment	0000001	INC	RD,RA	R[DR] ← R[SA] + 1	N, Z
Add	0000010	ADD	RD,RA,RB	R[DR] ← R[SA] + R[SB]	N, Z
Subtract	0000101	SUB	RD,RA,RB	R[DR] ← R[SA] − R[SB]	N, Z
Decrement	0000110	DEC	RD,RA	R[DR] ← R[SA] − 1	N, Z
AND	0001000	AND	RD,RA,RB	R[DR] ← R[SA] ∧ R[SB]	N, Z
OR	0001001	OR	RD,RA,RB	R[DR] ← R[SA] ∨ R[SB]	N, Z
Exclusive OR	0001010	XOR	RD,RA,RB	R[DR] ← $\overline{R[SA]}$ ⊕ R[SB]	N, Z
NOT	0001011	NOT	RD,RA	R[DR] ← $\overline{R[SA]}$	N, Z

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA: Instruction Specifications (continued)

Instruction Specifications for the Simple Computer - Part 2

Instruction	Opcode	Mnemonic	Format	Description	Status Bits
Move B	0001100	MOVB	RD,RB	R[DR] ← R[SB]	
Shift Right	0001101	SHR	RD,RB	R[DR] ← sr R[SB]	
Shift Left	0001110	SHL	RD,RB	R[DR] ← sl R[SB]	
Load Immediate	1001100	LDI	RD, OP	R[DR] ← zf OP	
Add Immediate	1000010	ADI	RD,RA,OP	R[DR] ← R[SA] + zf OP	
Load	0010000	LD	RD,RA	R[DR] ← M[SA]	
Store	0100000	ST	RA,RB	M[SA] ← R[SB]	
Branch on Zero	1100000	BRZ	RA,AD	if (R[SA] = 0) PC ← PC + se AD	
Branch on Negative	1100001	BRN	RA,AD	if (R[SA] < 0) PC ← PC + se AD	
Jump	1110000	JMP	RA	PC ← R[SA]	

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

ISA:Example Instructions and Data in Memory

Memory Representation of Instructions and Data

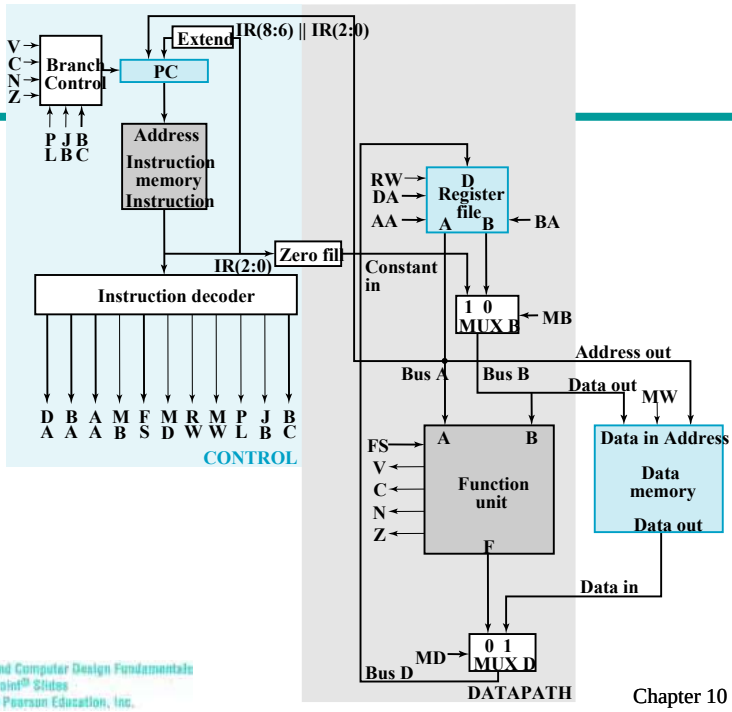
Deciimal Address	Memory Contents	Decimal Opcode	Other Fields	Operation
25	0000101 001 010 011	5 (Subtract)	DR:1, SA:2, SB:3	R1 ← R2 – R3
35	0100000 000 100 101	32 (Store)	SA:4, SB:5	M[R4] ← R5
45	1000010 010 111 011	66 (Add Immediate)	DR:2, SA:7, OP:3	R2 ← R7 + 3
55	1100000 101 110 100	96 (Branch on Zero)	AD: 44, SA:6	If R6 = 0, PC ← PC – 20
70	00000000011000000	Data = 192. After execution of instruction in 35, Data = 80.		

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

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 datapath from Figure 10-11 will be used
- The control unit will be defined as a part of the design
- The block diagram is shown on the next slide

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

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

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

PC Function

- PC function is based on instruction specifications involving jumps and branches taken from Slide 13:

Branch on Zero	BRZ	if (R[SA] = 0) PC ← PC + se AD
Branch on Negative	BRN	if (R[SA] < 0) PC ← PC + se AD
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

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

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

© 2004 Pearson Education, Inc.

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.

© 2004 Pearson Education, Inc.

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 in Slides 12 and 13.
- 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

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

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

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

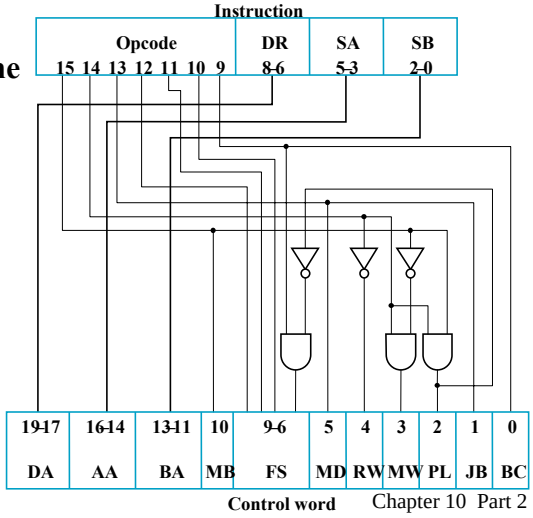
Instruction Decoder (continued)

- 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.

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

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



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Example Instruction Execution

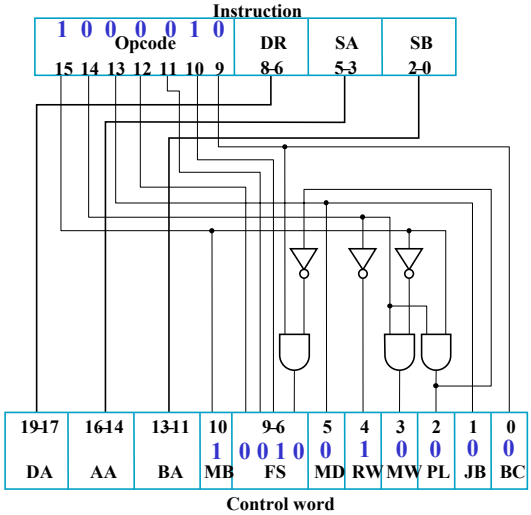
Six Instructions for the Single-Cycle Computer

Operation code	Symbolic name	Format	Description	Function	MB	MD	RW	MW	PL	JB	BC
1000010	ADI	Immediate	Add immediate operand	R[DR] ← R[SA] + zf I(2:0)	1	0	1	0	0	0	0
0010000	LD	Register	Load memory content into register	R[DR] ← M[R [SA]]	0	1	1	0	0	1	0
0100000	ST	Register	Store register content in memory	M [R [SA]] ← R[SB]	0	1	0	1	0	0	0
0001110	SL	Register	Shift left	R[DR] ← sl R [SB]	0	0	1	0	0	1	0
0001011	NOT	Register	Complement register	R[DR] ← $\overline{R[SA]}$	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] ≠ 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**

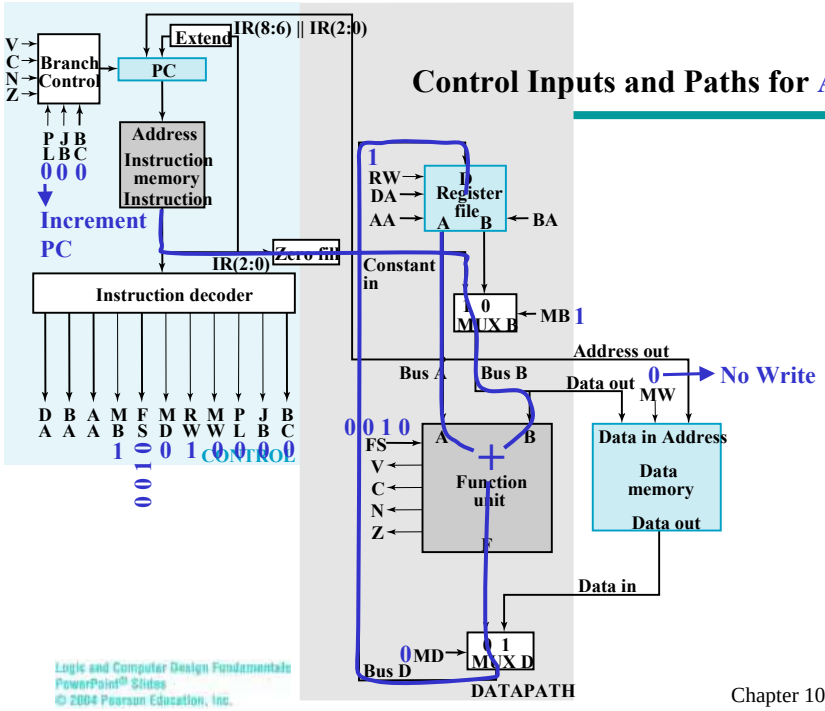
Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Decoding for ADI

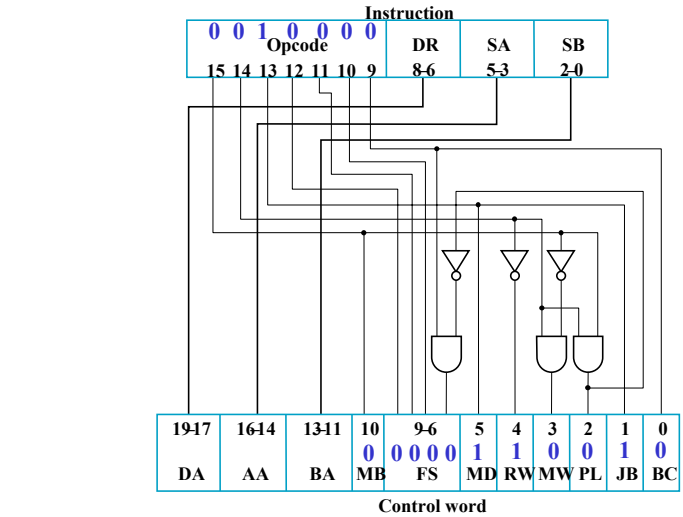


Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

Control Inputs and Paths for ADI

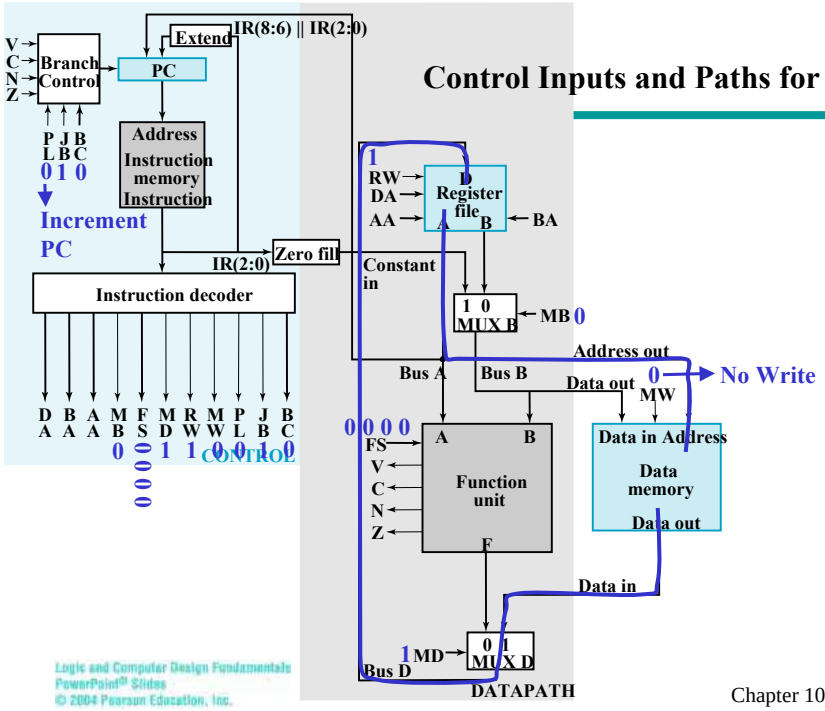


Decoding for LD

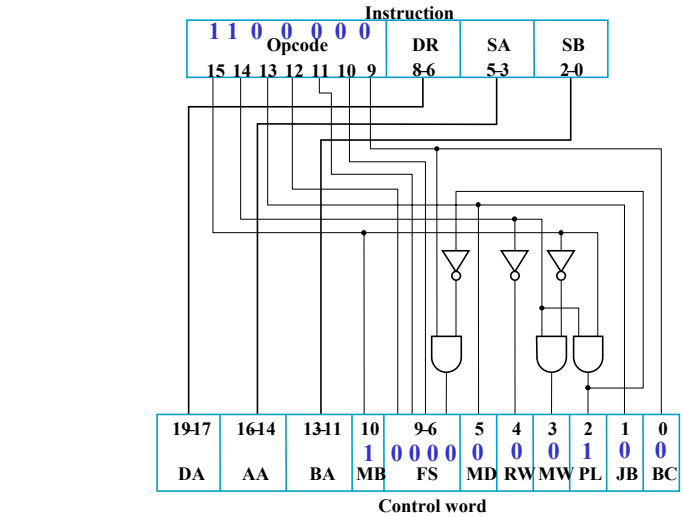


Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

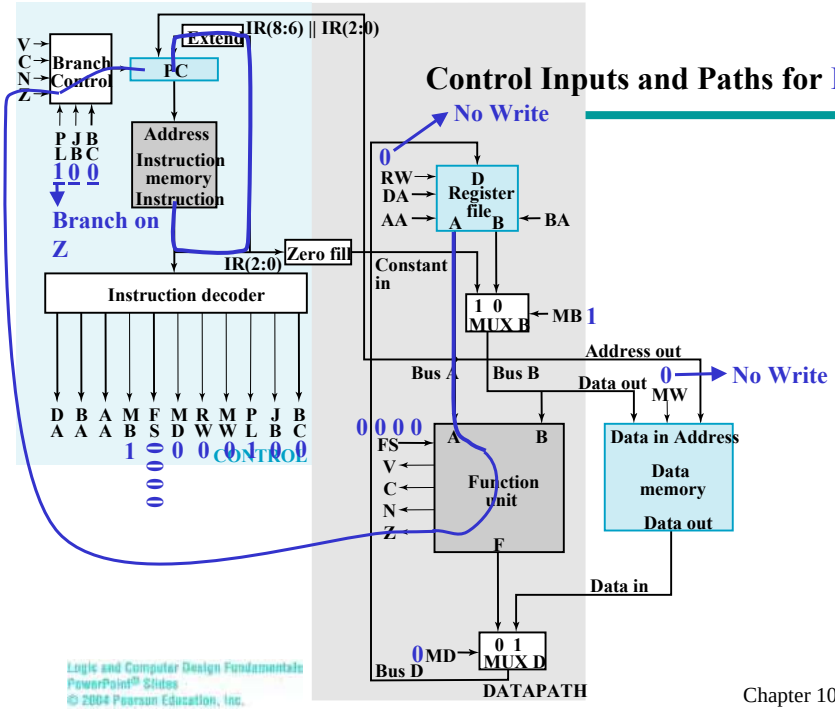
Control Inputs and Paths for LD



Decoding for BRZ



Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.



Terms of Use

- © 2004 by Pearson Education,Inc. All rights reserved.
- The following terms of use apply in addition to the standard Pearson Education Legal Notice.
- Permission is given to incorporate these materials into classroom presentations and handouts only to instructors adopting Logic and Computer Design Fundamentals as the course text.
- Permission is granted to the instructors adopting the book to post these materials on a protected website or protected ftp site in original or modified form. All other website or ftp postings, including those offering the materials for a fee, are prohibited.
- You may not remove or in any way alter this Terms of Use notice or any trademark, copyright, or other proprietary notice, including the copyright watermark on each slide.
- Return to Title Page

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.