Computer Design Basics

CO 2206 Computer Organization

Topics

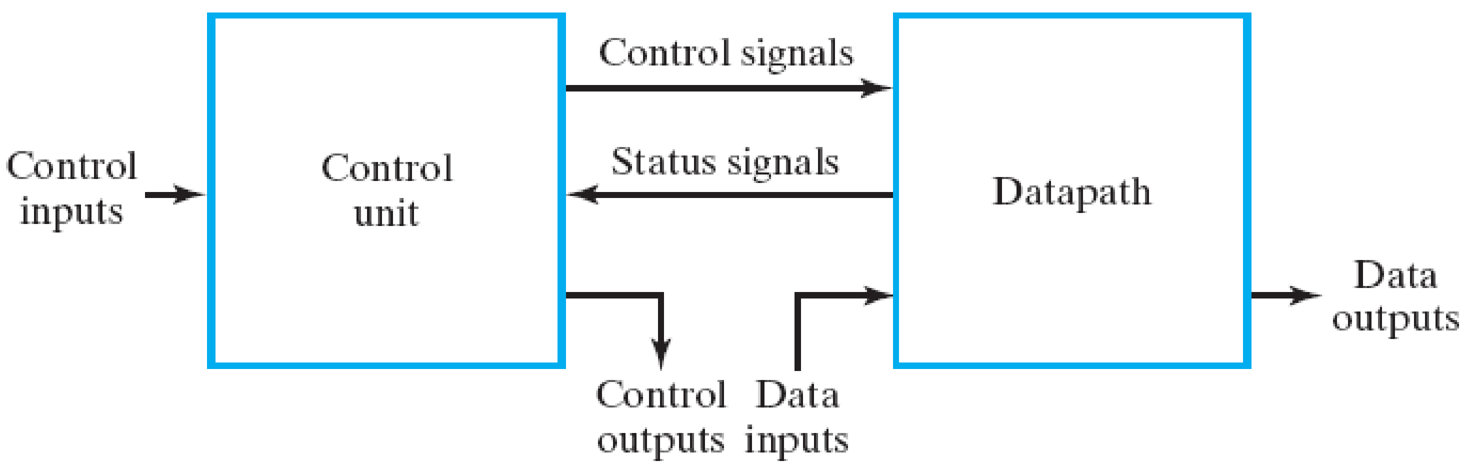
- Digital Systems – Datapath and Control Unit
- Datapath
  - Arithmetic Logic Unit
  - Shifter
  - Function Unit
  - Control Word
- Control Unit
  - Instruction Set Architecture
  - Instruction Format
  - Instruction Specification
  - Single-cycle Hardwired Control
  - Instruction Decoder
- Single Cycle Computer Issues

Digital Systems: Modular Design

- Digital systems are designed using a modular, hierarchical approach
  - The system is partitioned into subsystems or modules
  - The modules are constructed hierarchically from functional blocks e.g. registers, counters, decoders, primitive gates, etc
  - Subsystems communicate with data and control signals

Digital Systems: Subsystems

- Generally partition digital system into
  - *Datapath*, which performs data-processing operations
  - Control unit, which determines the sequence of those operations



Subsystem: Control Unit

- Control signals activate the various data-processing operations
  - To activate a sequence of operations, the control unit sends the proper sequence of control signals to the datapath
  - Control unit receives status bits from the datapath to determine the next sequence of operations to be performed
- Datapath and control unit may also interact with memory, IO logic, etc

Subsystem: Datapath

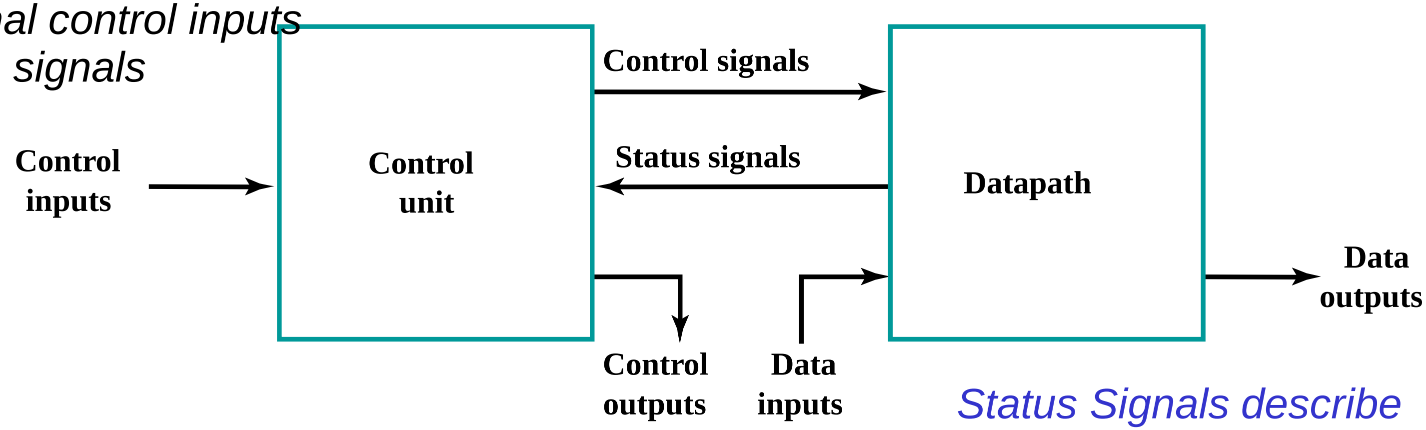
- Datapath is defined by their registers and operations performed on data stored
- Examples of register operations are
  - load,
  - clear,
  - shift and
  - count
- The movement of data stored in registers and processing performed on the data are referred as *register transfer operations*

Datapath and Control

- Datapath and control unit are the 2 parts of the processor or CPU:
  - Datapath - performs data transfer and processing operations
  - Control Unit - Determines the enabling and sequencing of the operations

The control unit sends:
- Control signals
- Control outputs

The control unit receives:
- External control inputs
- Status signals

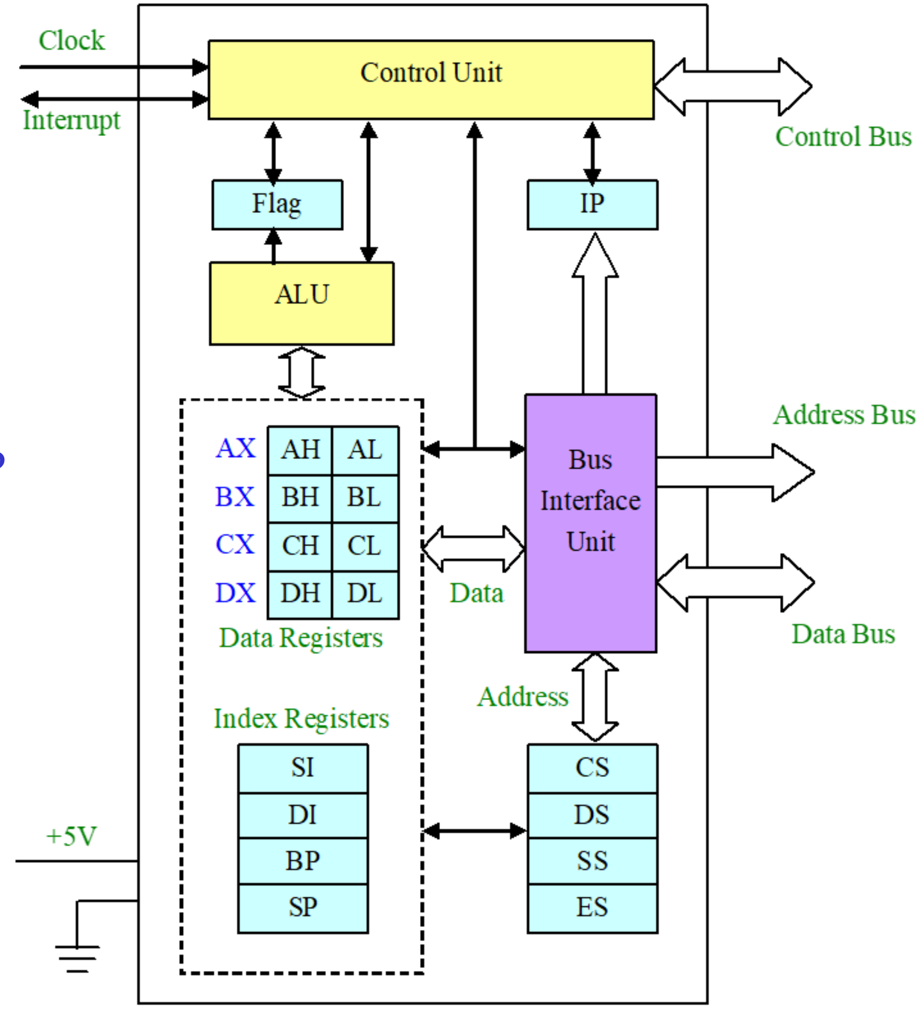


Status Signals describe properties of the state of the datapath

CPU Recall: 8086 Architecture

Intel 8086

Internal Block Diagram



Identify Control and Datapath?

Datapath

Datapath – 1

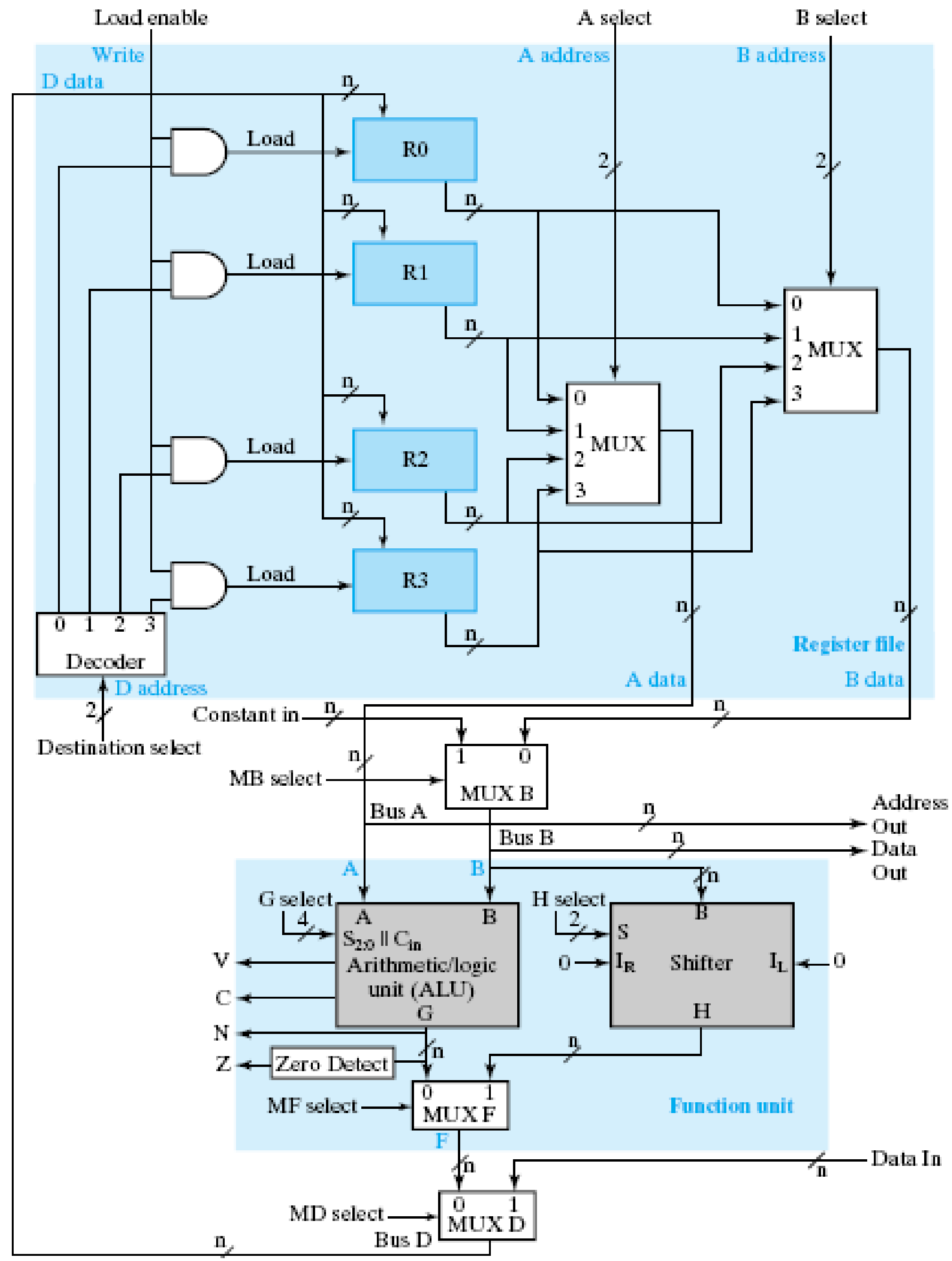
- In addition to the registers, the datapath contains the digital logic that implements the various microoperations
- The datapath includes registers, selection logic for registers, ALU, shifter, multiplexers
- The control unit directs the information flow by applying signals to the select inputs

Datapath - 2

- Computer systems often employ a no. of registers in conjunction with a shared ALU
- To perform a microoperation
  - The contents of specified source registers are applied to the inputs of the ALU
  - ALU performs an operation
  - Result transferred to a destination register
- With ALU as combinational circuit, the entire register transfer operation is performed during one clock cycle

- CPU Datapath Example:

We will build this!

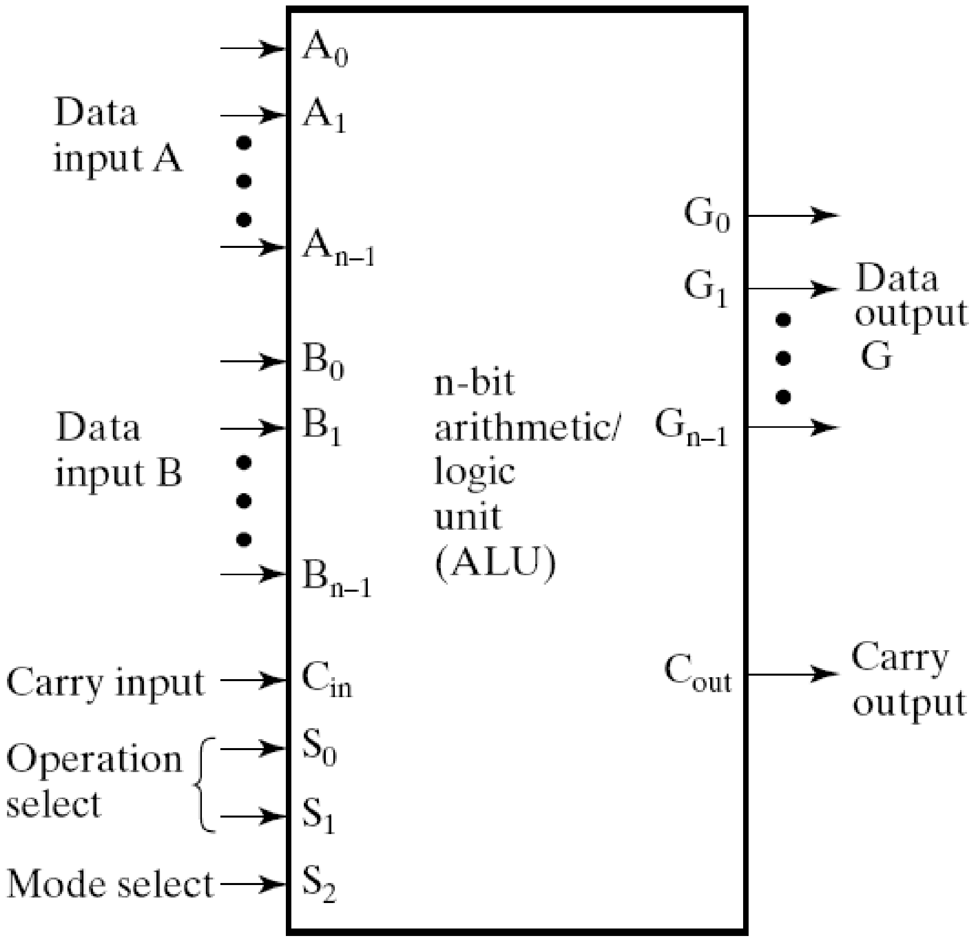


Arithmetic Logic Unit

- ALU is a combinational circuit that performs a set of basic arithmetic and logic microoperations
  - Has a no. of selection lines used to determine the operation to be performed
  - k selection lines can specify up to 2^k distinct operations

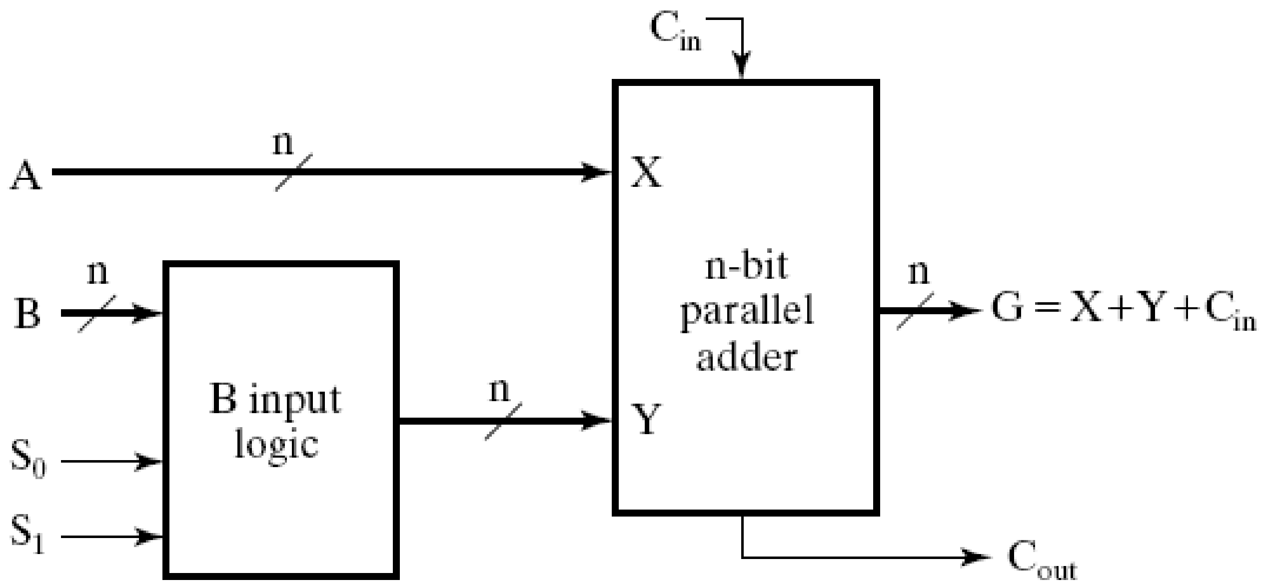
ALU: Function Selection

- $S_2 = 0$
  - Arithmetic operations
  - S_0, S_1, C_{in} specify 8 arithmetic operations
- $S_2 = 1$
  - S_0 and C_{in} specify 4 logic operations



ALU – Arithmetic Circuit

- Basic component of an arithmetic circuit is a parallel adder
  - Constructed with a no. of cascading FA
  - Value of Y is controlled by S_0, S_1



Arithmetic: Function Table

Function Table for Arithmetic Circuit

Select		Input	$G = A + Y + C_{in}$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0's	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1's	$G = A - 1$ (decrement)	$G = A$ (transfer)

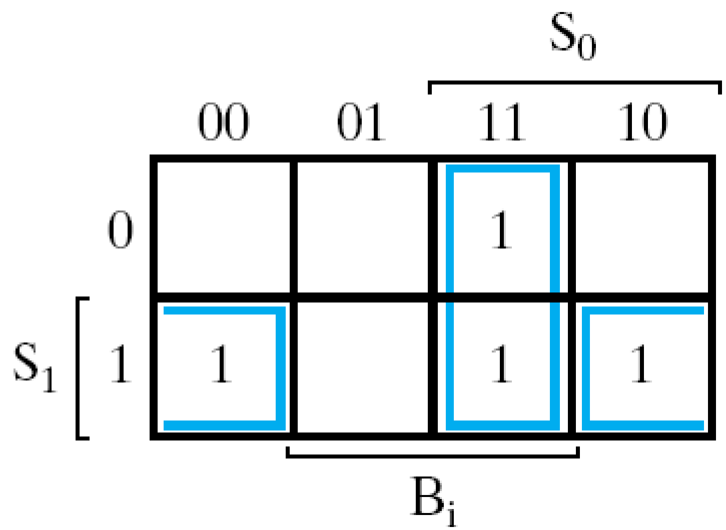
Arithmetic: Implement Selection

- B input logic can be implemented with
  - *n* multiplexers
    - Data inputs to each multiplexer in stage *i* are 0, B_i, B_i' and 1 (as per Y), corresponding to S_1S_0
    - thus, arithmetic circuit can be constructed with *n* FA and *n* 4x1 mux
  - *n* combinational circuit
    - Reduced no. of gates

Arithmetic: Selection K-Map

Inputs			Output	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \overline{B_i}$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

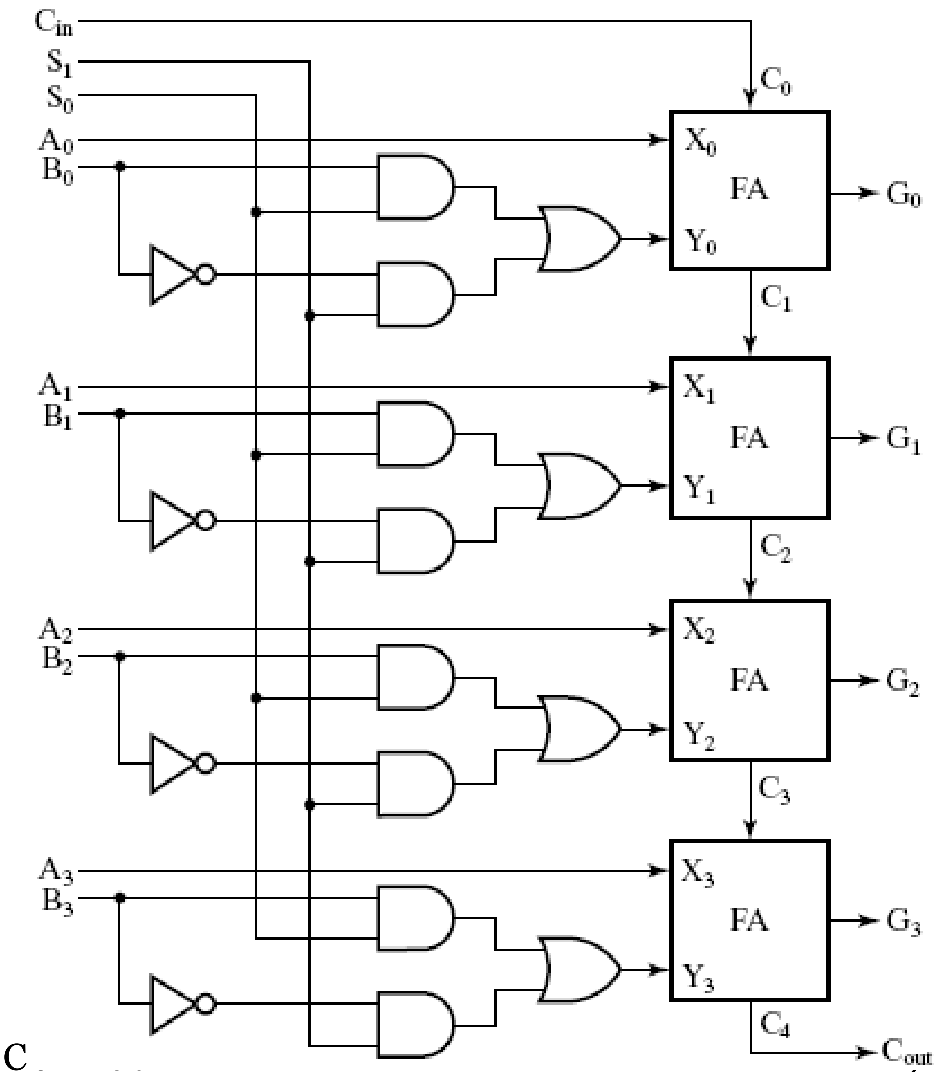
(a)Truth table



(b) Map Simplification:
$Y_i = B_i S_0 + \overline{B_i} S_1$

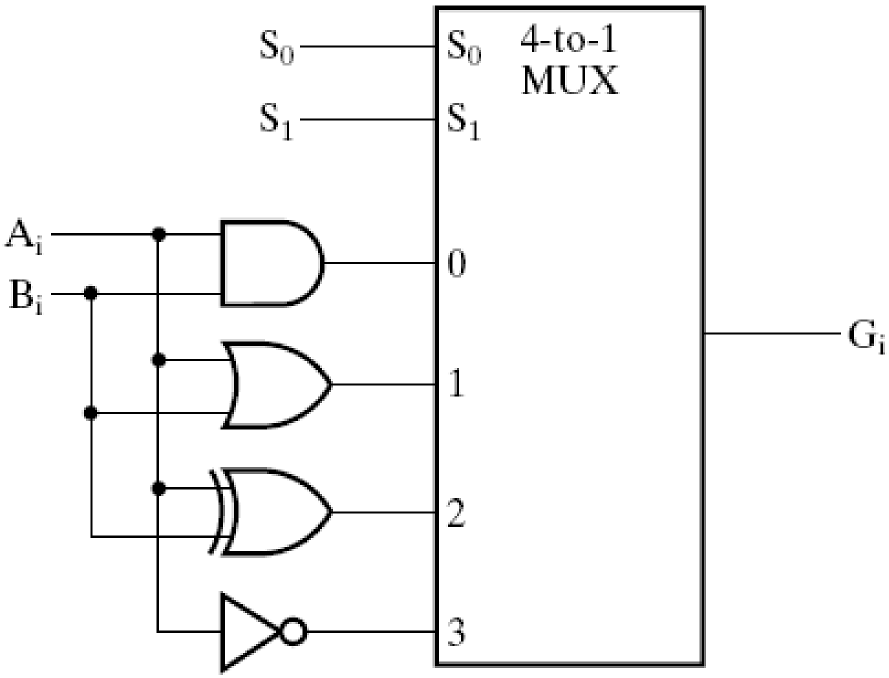
Arithmetic: Selection Circuit

- Logic diagram of 4-bit arithmetic circuit



ALU – Logic Circuit

- One stage (cell) of logic circuit: MUX to select function



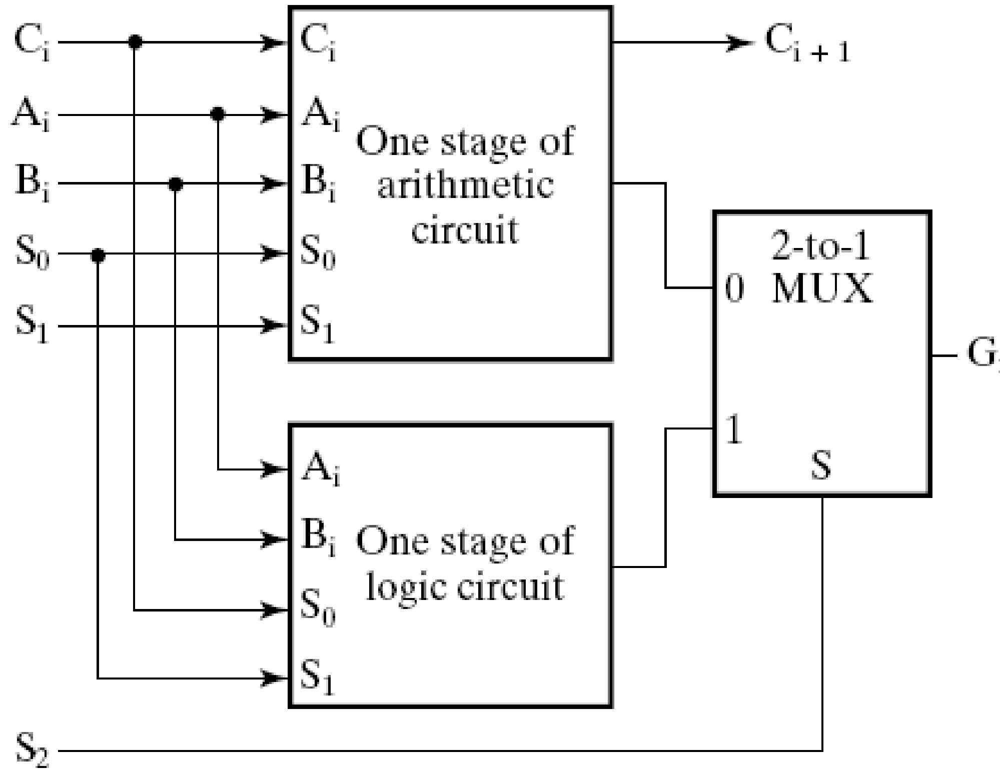
(a) Logic Diagram

S_1	S_0	Output	Operation
0	0	$G = A \wedge B$	AND
0	1	$G = A \vee B$	OR
1	0	$G = A \oplus B$	XOR
1	1	$G = \overline{A}$	NOT

(b) Function Table

ALU – Arithmetic & Logic

- Note that S_0 of one stage logic circuit is connected to C_i, and S_0 to S_1: strange connection to provide more systematic encoding of the control variables when the shifter is added later



ALU: Selection

- $S_2 = 0$,
  - Arithmetic operations selected
- $S_2 = 1$,
  - Logic operations selected
- S_0,S1 and C_{in} control the selection of arithmetic and logic operations

ALU: Full Function Table

Function Table for ALU

Operation Select					
S_2	S_1	S_0	C_{in}	**Operation**	**Function**
0	0	0	0	$G \leftarrow A$	Transfer *A*
0	0	0	1	$G \leftarrow A + 1$	Increment *A*
0	0	1	0	$G \leftarrow A + B$	Addition
0	0	1	1	$G \leftarrow A + B + 1$	Add with carry input of 1
0	1	0	0	$G \leftarrow A + \overline{B}$	*A* plus 1's complement of *B*
0	1	0	1	$G \leftarrow A + \overline{B} + 1$	Subtraction
0	1	1	0	$G \leftarrow A + 1$	Decrement *A*
0	1	1	1	$G \leftarrow A$	Transfer *A*
1	X	0	0	$G \leftarrow A \wedge B$	AND
1	X	0	1	$G \leftarrow A \vee B$	OR
1	X	1	0	$G \leftarrow A \oplus B$	XOR
1	X	1	1	$G \leftarrow \overline{A}$	NOT (1's complement)

Shifter

- To shift an operand by *m* bit positions, a shifter must perform a series of *m* 1-bit position shifts
  - Taking *m* clock cycles
- In datapath applications, data often must be shifted more than 1 bit position in a single clock cycle
- A *barrel shifter* is a combinational circuit that shifts or rotates in one clock cycle

Barrel Shifter

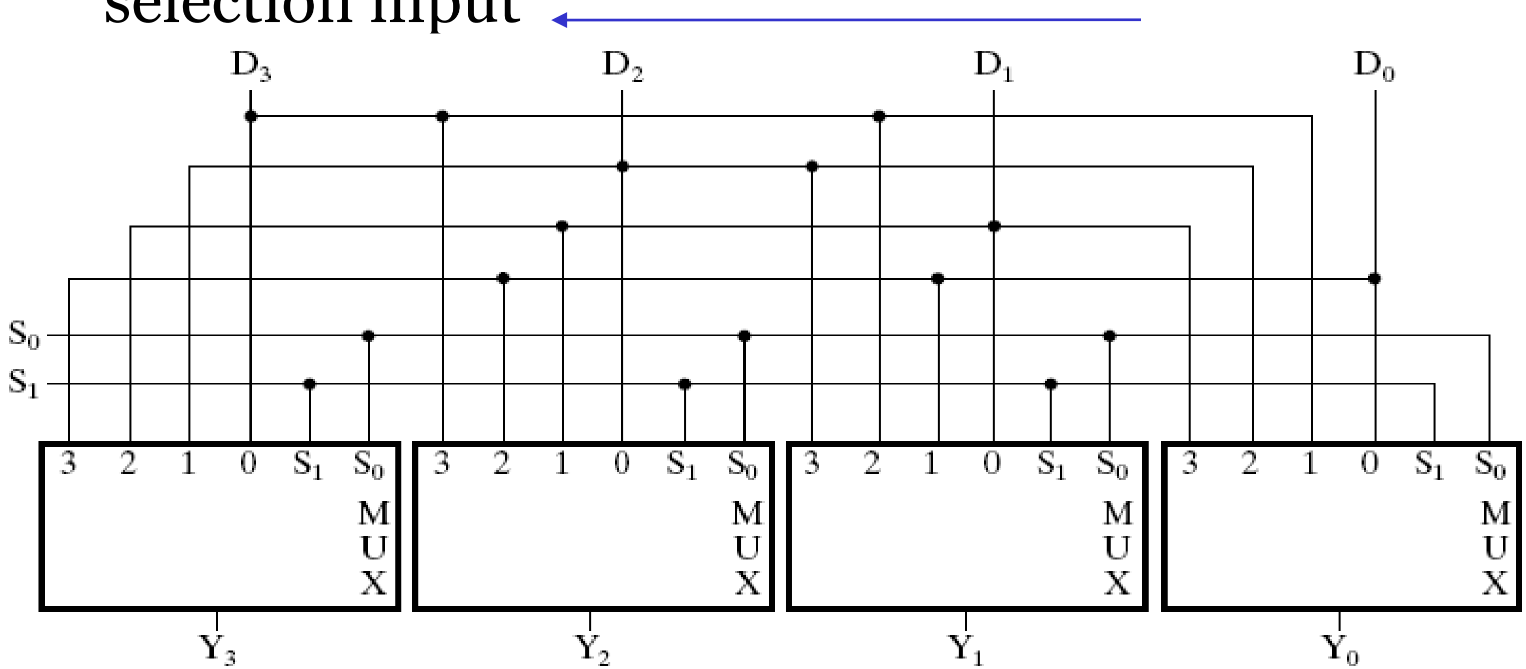
- A barrel shifter can rotate left and right
  - e.g. a left rotation by 3 position is same as right rotation by 1 position (for 4-bit shifter)
  - In a 2^n-bit barrel shifter, i position of left rotation is same as $2^n - i$ bits of right rotation

Function Table for 4-Bit Barrel Shifter

Select		Output				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	Operation
0	0	D_3	D_2	D_1	D_0	No rotation
0	1	D_2	D_1	D_0	D_3	Rotate one position
1	0	D_1	D_0	D_3	D_2	Rotate two positions
1	1	D_0	D_3	D_2	D_1	Rotate three positions

Barrel Shifter

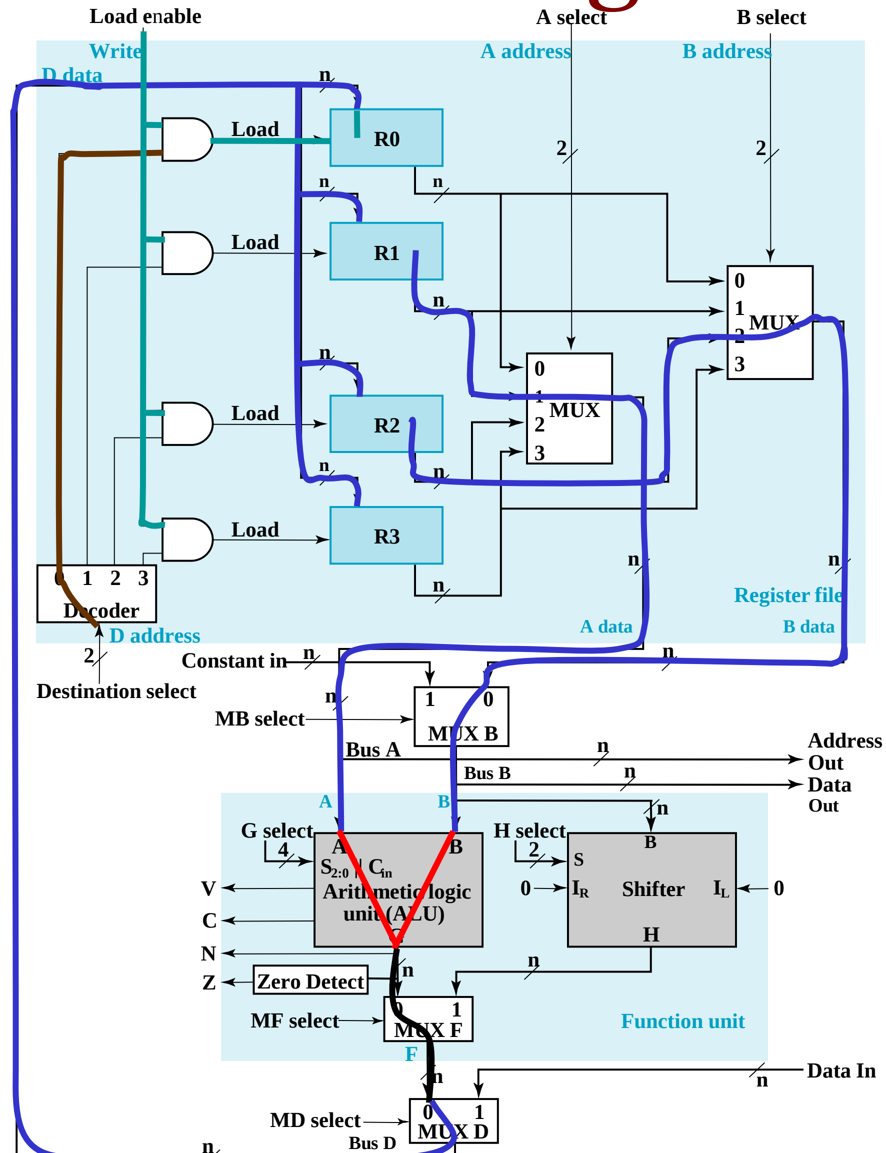
- A barrel shifter with 2^n input and output lines requires
  - 2^n multiplexers, each with 2^n data inputs and n selection input



Datapath Example: Performing a Microoperation

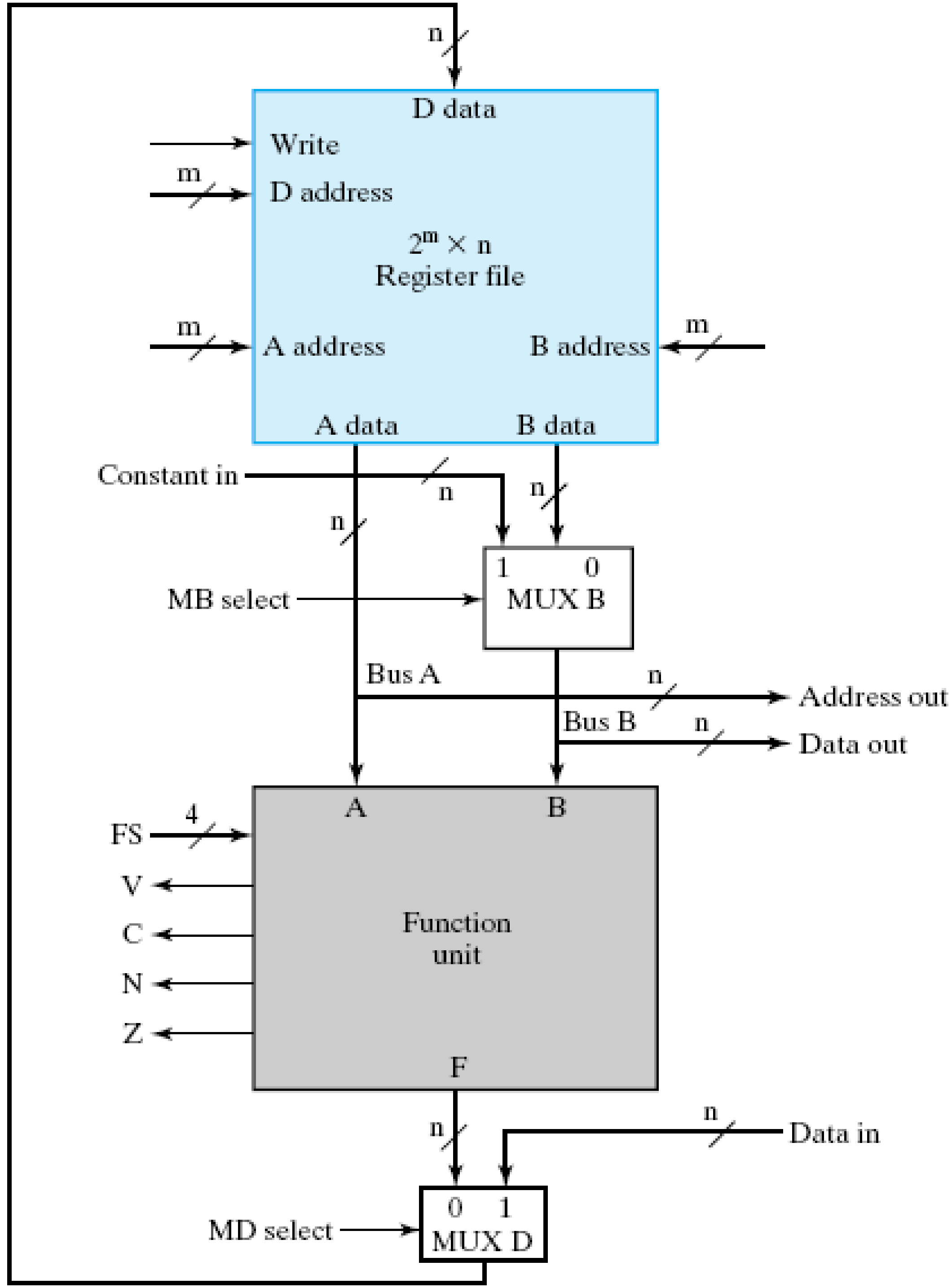
Microoperation: R0 ← R1 + R2

- 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 Representation: Register File - 1

- A set of registers may be organised into a *register file (register array)*
- A typical register file permits one or more words to be read and written, all simultaneously
  - *A address* access a word (register) to be read onto *A data*
  - *B address* access a word (register) to be read onto *B data*
  - *D address* access a word (register) to be written into from *D data*
- All register access occur in the same clock cycle
- Size of register file is 2^m x n
  - m is the no. of bits used to identify the register
  - n is the no. of bits per register (word size)


n
D data
Write
m
D address
2^m × n
Register file
m
A address
B address
m
A data
B data
Constant in
n
n
n
1
0
MUX B
MB select
Bus A
n
Address out
Bus B
n
Data out
A
B
FS
4
V
C
N
Z
Function
unit
F
n
n
Data in
0
1
MUX D
MD select

Function Unit

- *Function unit* is formed by grouping the shifter, ALU and multiplexer MUX F
- Function unit has 4 status bits
  - *V* (overflow), *C* (carry), *N* (sign), *Z* (zero)
- *FS* input selection bits determine the microoperation to be carried out by the function unit
- *MF*, *G* and *H* select are determined from *FS*
  - MF = 1, when FS(3:2) = 11
    - H determines the shift function
  - When MF = 0,
    - G determines the ALU function

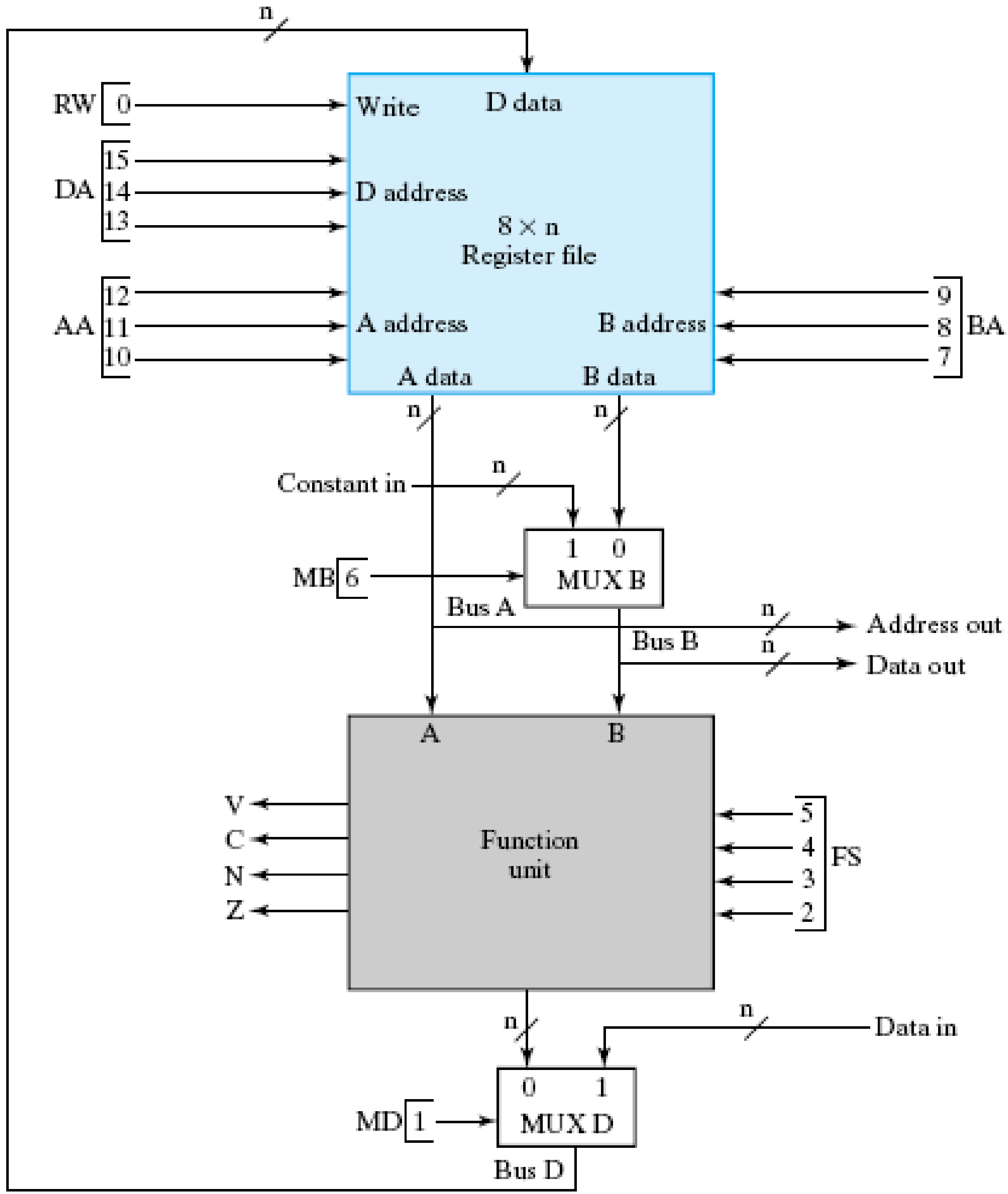
Function Unit: Function Table

***G* Select, *H* Select, and *MF* Select Codes Defined in Terms of *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	$F \leftarrow \text{sl } B$

An Example: Micro-coding

- Consider a register file with 8 registers R0 to R7
- Register file inputs to function unit through Bus A and Bus B
- MUX B selects between constant values or register values on B data
- MUX D selects the function unit output or data on Data in as input for register file



Control signals are numbered as bit position; giving a ***16-bit Control Word***

Control Word: Bit Assignment

- The 16-bit control word specifies the microoperation
  - DA, AA, BA selects the registers
  - MB determines the selection in MUX B
  - FS controls the operation of function unit
  - MD determines the selection in MUX D
  - RW determines write on register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DA			AA			BA			MB	FS				MD	RW

Control Word: Encoding

Encoding of Control Word for the Datapath

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			$F \leftarrow A + B + 1$	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				

Control Word: Example Microop

Examples of Microoperations for the Datapath, Using Symbolic Notation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	$R1$	$R2$	$R3$	Register	$F = A + \overline{B} + 1$	Function	Write
$R4 \leftarrow \text{sl } R6$	$R4$	—	$R6$	Register	$F = \text{sl } B$	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

Many microoperations can be performed by the same datapath

Control Word: Example Encoding

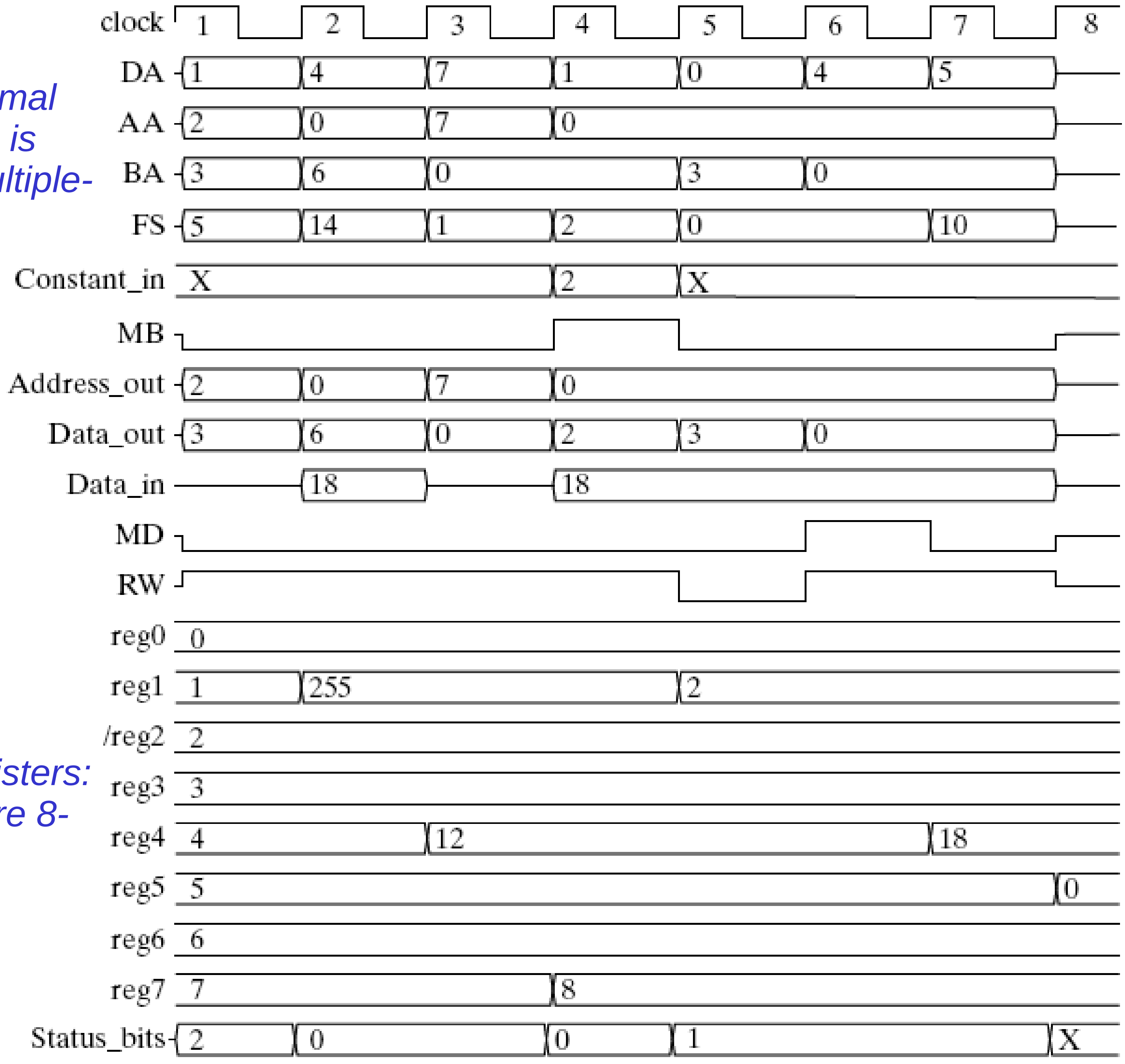
Examples of Microoperations from Table 10-6, Using Binary Control Words

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	0101	0	1
$R4 \leftarrow \mathrm{sl}\ \mathrm{R6}$	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
$\mathrm{Data\ out} \leftarrow R3$	XXX	XXX	011	0	XXXX	X	0
$R4 \leftarrow \mathrm{Data\ in}$	100	XXX	XXX	X	XXXX	1	1
$R5 \leftarrow 0$	101	000	000	0	1010	0	1

Sequences of microoperations can be realized by providing a control unit that produces the appropriate sequences of control words

Changes in registers appear in the clock cycle after the microoperation is specified

Unsigned decimal representation is used for all multiple-bit signals



Eight 8-bit registers: reg0 to reg7 are 8-bit registers

owh@ieee.org

Control Unit

A Simple Computer Architecture - 1

- A portion of input to the processor consists of a sequence of *instructions*
- Each instruction specifies
  - The operation to perform
  - Operands to use
  - Where to place the result
  - Which instructions to execute next, in some cases
- Instructions are usually stored in memory

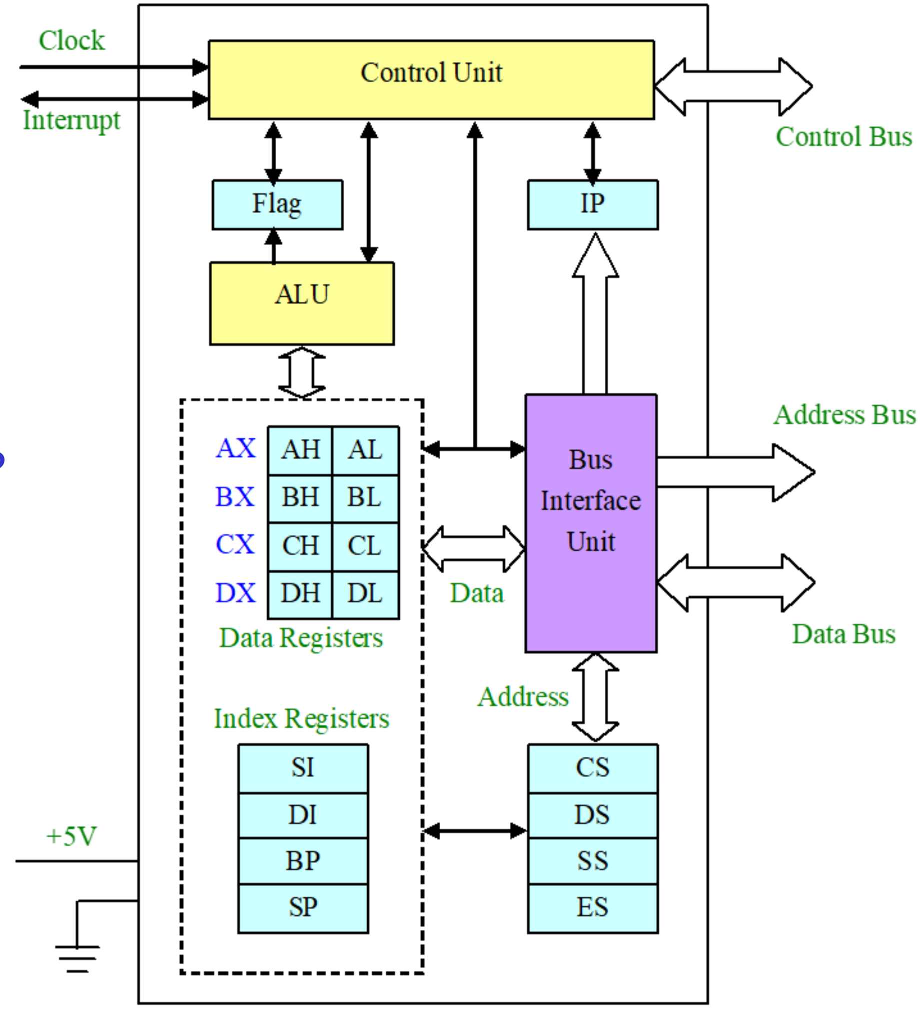
A Simple Computer Architecture - 2

- Memory address of the instruction to be executed comes from the *PC* register
  - PC has logic for counting
  - PC needs parallel load capability
- Thus, the Control Unit contains a PC and necessary logic to interpret instruction
- *Executing* an instruction means activating the necessary sequence of microoperations in the datapath

CPU Recall: 8086 Architecture

Intel 8086

Internal Block Diagram



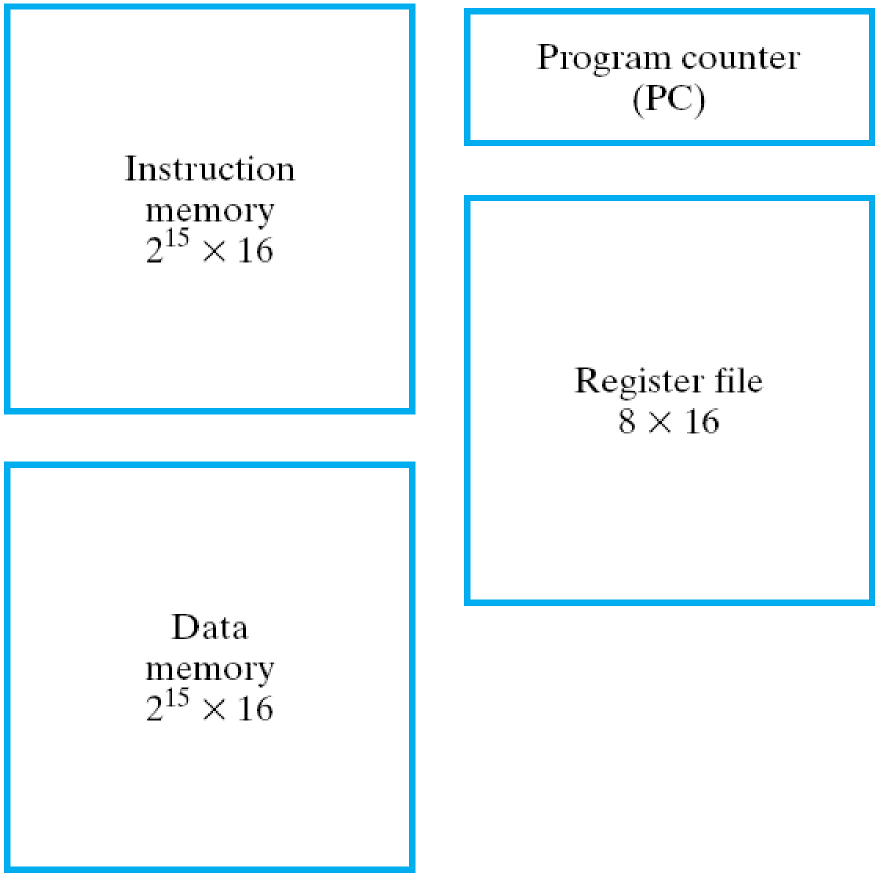
Identify Control and Datapath?

Instruction Set Architecture

- A collection of instructions for a computer is the *instruction set*
- An *instruction set architecture* consists of
  - Storage resources - instructions and data are usually stored together in the same memory
  - Instruction formats
  - Instruction specifications
- The Instruction Set Architecture (ISA) is the part of the processor that is visible to the programmer or compiler writer

Storage Resources

- Specifies the resources the user sees available for storing information



Quotes from the Art of Assembly Language by Randy Hyde - 1

The instruction set architecture (or ISA) is one of the most important design issues that a CPU designer must get right from the start. Features like caches, pipelining, superscalar implementation, etc., can all be grafted on to a CPU design long after the original design is obsolete. However, it is very difficult to change the instructions a CPU executes once the CPU is in production and people are writing software that uses those instructions. Therefore, one must carefully choose the instructions for a CPU.

Quotes from the Art of Assembly Language by Randy Hyde - 2

An instruction set, or instruction set architecture (ISA), is the part of the computer architecture related to programming, including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O. An ISA includes a specification of the set of opcodes (machine language), the native commands implemented by a particular CPU design.

Quotes from the Art of Assembly Language by Randy Hyde - 3

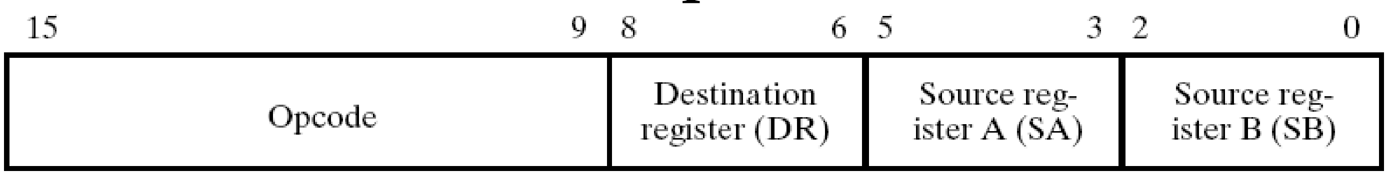
Instruction set architecture is distinguished from the microarchitecture, which is the set of processor design techniques used to implement the instruction set. Computers with different microarchitectures can share a common instruction set. For example, the Intel Pentium and the AMD Athlon implement nearly identical versions of the x86 instruction set, but have radically different internal designs.

Instruction Formats - 1

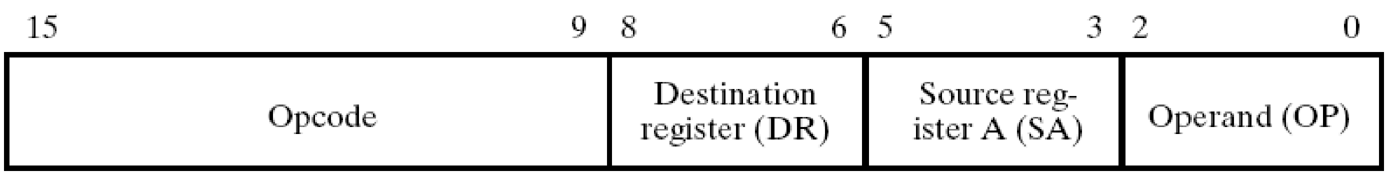
- An instruction is divided into fields:
  - Opcode
  - Operand(s)/Others
- *Opcode* is a field that specifies an operation e.g. add, subtract, shift.
  - No. of bits required for opcode is a function of the total no. of operations in the ISA
  - Specific bit combination is assigned to each operation

Instruction Formats - 2

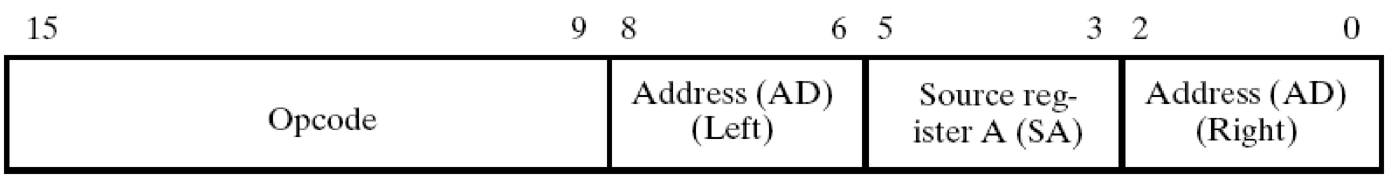
- An instruction must also specifies
  - the registers or memory words in which the operands are to be found
  - where the result is to be placed



(a) Register



(b) Immediate



(c) Jump and Branch

Instruction Formats - 3

- In (a), 3 or fewer registers are specified
  - For some operation (by design), e.g. store to memory
    - SA specify the register that contains the memory address
    - SB specify the register that contains the data to be stored
    - DR has no effect (this will prevent writing to register file)
- In (b), the operand is specified in the instruction
- (c) does not change any register or memory contents

Instruction Specifications

- For each instruction, the opcode can be represented using mnemonics, and all field specified in non-binary format
- An *assembler* is then used to convert this representation to its binary equivalent

Instruction Specifications for the Simple Computer

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	$R[DR] \leftarrow \text{sr } R[SB]$	
Shift Left	0001110	SHL	RD,RB	$R[DR] \leftarrow \text{sl } R[SB]$	
Load Immediate	1001100	LDI	RD, OP	$R[DR] \leftarrow \text{zf } OP$	
Add Immediate	1000010	ADI	RD,RA,OP	$R[DR] \leftarrow R[SA] + \text{zf } OP$	
Load	0010000	LD	RD,RA	$R[DR] \leftarrow M[SA]$	
Store	0100000	ST	RA,RB	$M[SA] \leftarrow R[SB]$	
Branch on Zero	1100000	BRZ	RA,AD	if $(R[SA] = 0)$ $PC \leftarrow PC + \text{se } AD$	
Branch on Negative	1100001	BRN	RA,AD	if $(R[SA] < 0)$ $PC \leftarrow PC + \text{se } AD$	
Jump	1110000	JMP	RA	$PC \leftarrow R[SA]$	

Example: Instruction Representation

Memory Representation of Instructions and Data

Decimal Address	Memory Contents	Decimal Opcode	Other Fields	Operation
25	0000101 001 010 011	5 (Subtract) (register format)	DR:1, SA:2, SB:3	R1 ← R2 – R3
35	0100000 000 100 101	32 (Store) (register format)	SA:4, SB:5	M[R4] ← R5
45	1000010 010 111 011	66 (Add Immediate) (immediate format)	DR:2, SA:7, OP:3	R2 ← R7 + 3
55	1100000 101 110 100	96 (Branch on Zero) (jump and branch format)	AD: 44, SA:6	If R6 = 0, PC ← PC – 20
70	0000000001100000	Data = 192. After execution of instruction in 35, Data = 80. Assume R4=70, R5=80 before instruction in 35		

Example: Some Explanation

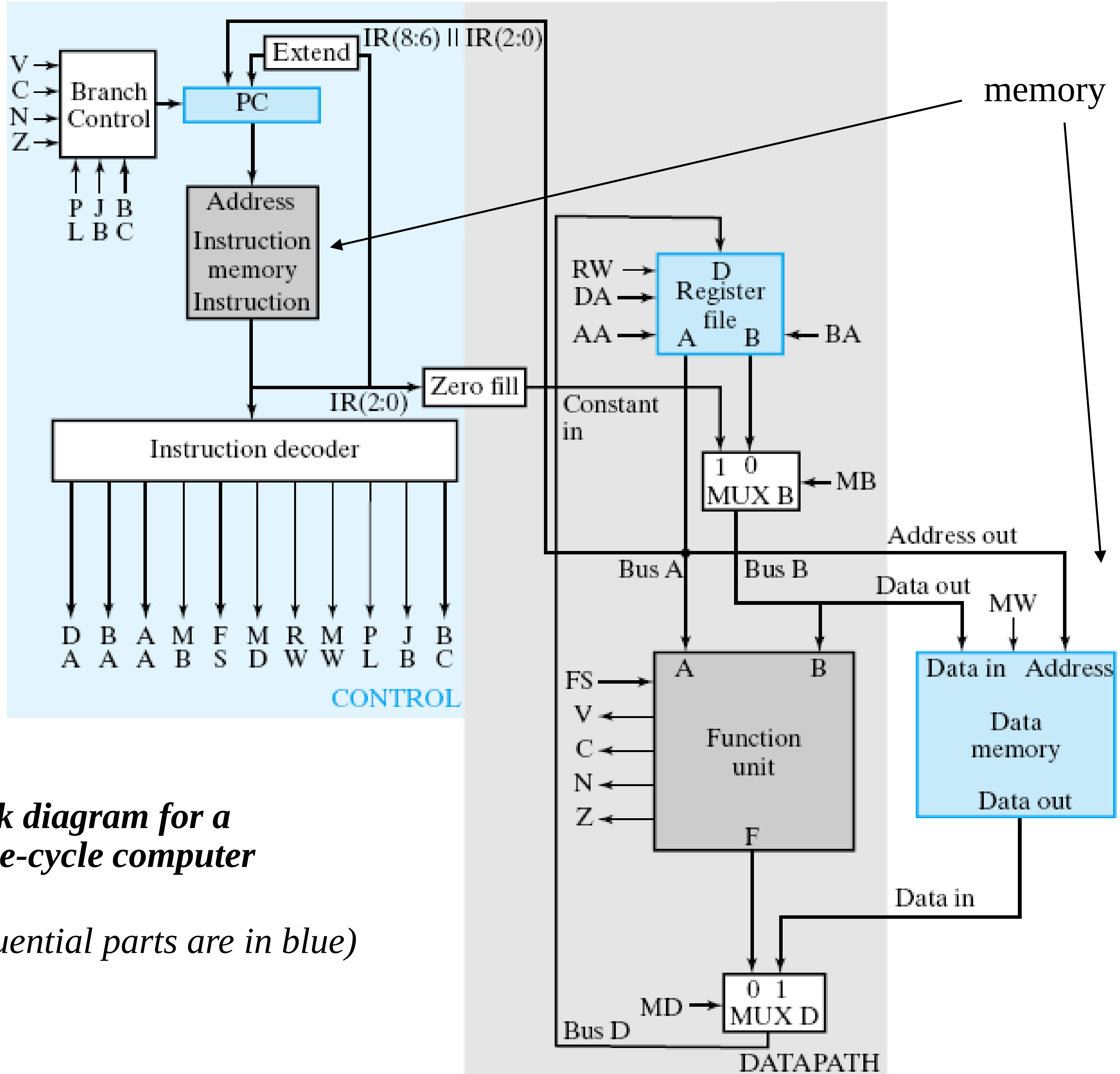
- In location 55,
  - AD (Left) = 101, AD (Right) = 100
    - AD = 101100 (left most is 1, i.e. -ve)
  - Combine and sign-extended (to 16-bit) = 1111111111101100 = -20 in 2's complement
  - Assume that addition to PC occurs before PC has been incremented
    - If R6 = 0, next instruction = PC-20 = 35
    - IF R6 ≠ 0, next instruction = PC = 56

Instruction to Microoperation

- Control unit uses address provided by PC to retrieve instruction from memory
  - Decodes the opcode and other instruction fields to perform required microoperations
- A **microoperation** is specified by control word in the h/w and decoded by the h/w
- A computer **operation** often requires a sequence of microoperations

Single-cycle Hardwired Control - 1

- For convenience, instruction memory is included with the control unit
- Instruction memory output goes to
  - Instruction decoder
  - *Extend* provide the address offset to PC
    - Extension appends the leftmost bit of 6-bit AD address offset
    - Sign extension to preserve 2's complement representation i.e. if leftmost bit = 1, append all 1's; otherwise all 0's



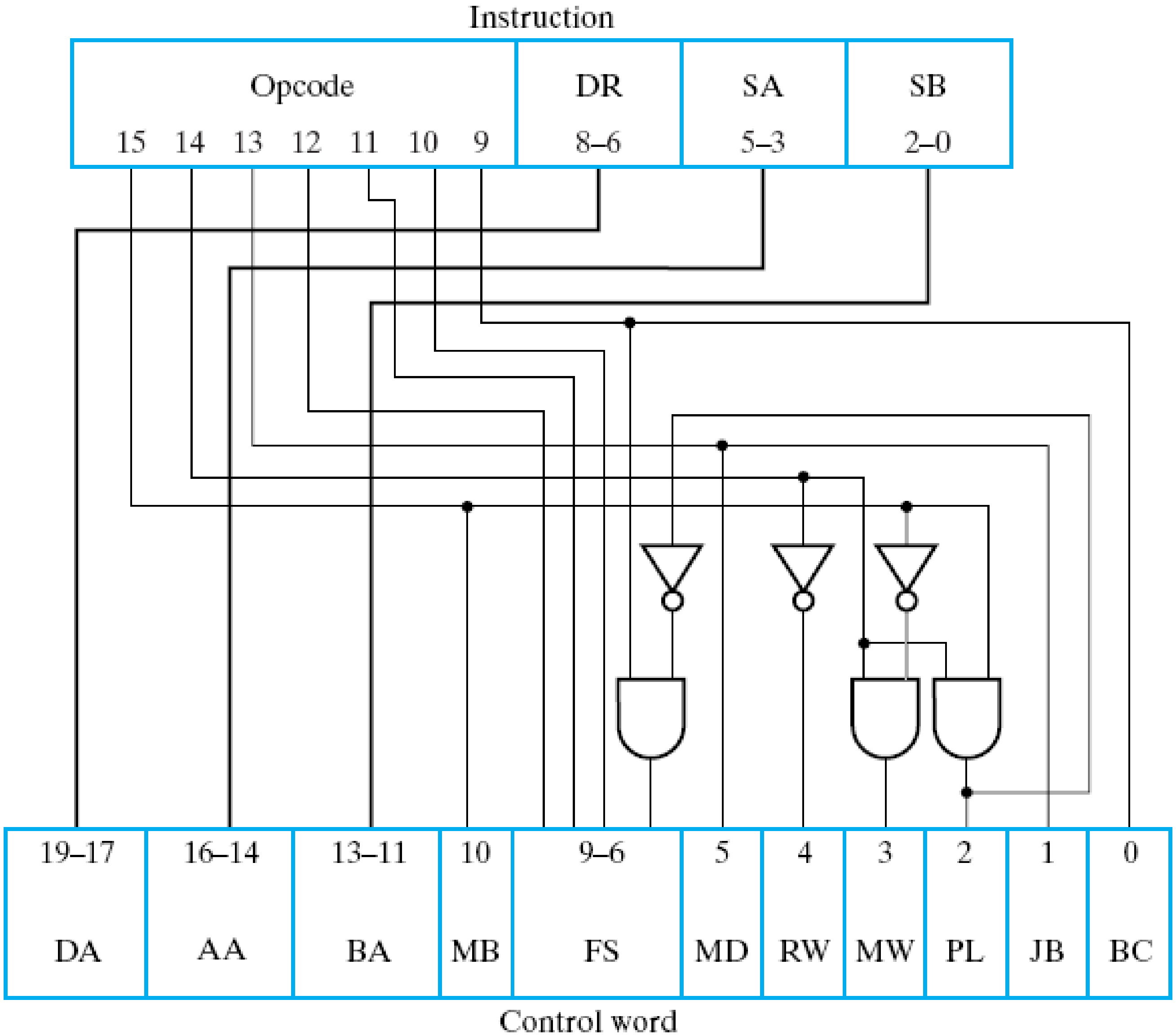
Block diagram for a single-cycle computer

(Sequential parts are in blue)

Single-cycle Hardwired Control - 2

  - *Zero Fill* provide the constant input to the datapath (for immediate format instructions)
    - Appends 13 0's to the left of the operand field to form 16-bit unsigned operand
    - E.g. 110 becomes 0000000000000110 (+6)
- PC is updated in each clock cycle
  - PC is a complex register
  - behaviour depends on opcode, N and Z
- Single-cycle computer obtains and executes an instruction all in a single clock cycle

Instruction Decoder - 1



owh@ieee.org

Instruction Decoder - 2

- Instruction decoder
  - A combinational circuit
  - Provides all control words for the datapath
    - Based on the contents of the fields of the instruction
- A no. of fields of the control word comes directly from the instruction
  - E.g. DA, AA, BA = DR, SA, SB, respectively
- Other bits are implemented through logic
  - By careful divide of instructions into function types

Truth Table for Instruction Decoder Logic: Function Division

	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

Single-Cycle Computer Issues - 1

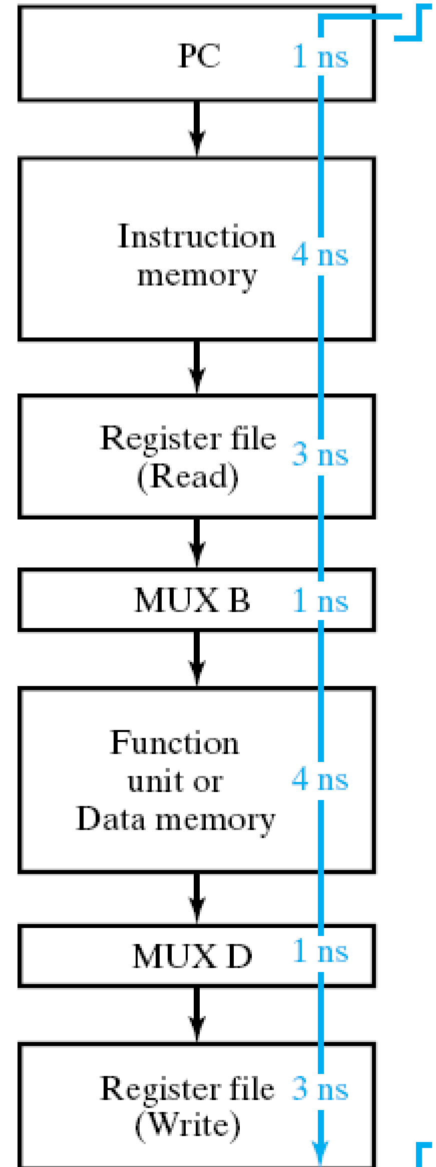
- Complex operations cannot be accomplished in a single clock cycle
- Single-cycle computer uses 2 distinct memories – one for instruction and one for data
  - If both instructions and data are in the same memory, 2 read accesses are required
    - First to obtain the instruction
    - Second, if required, to read/write the data word

Single-Cycle Computer Issues - 2

  - Since 2 different addresses must be applied to the memory address inputs, at least 2 clock cycles are required to obtain and execute the instruction
  - Multiple-cycle computer uses a single memory
- Has a lower limit on the clock period based on the worst delay path

Single-Cycle Computer Issues - 3

- If the worst delay is 17ns, this limits the clock frequency to 58.8MHz
  - Too slow for modern computer CPU
- To have a higher clock frequency
  - Reduce the delays of the components on the path or
  - Reduce the number of components on the path



Summary - 1

- Digital systems can generally be partitioned into two sections: Datapath and Control unit
- The datapath includes registers, selection logic for registers, ALU, shifter, multiplexers
- ALU is a combinational circuit that performs a set of basic arithmetic and logic microoperations
- A barrel shifter is a combinational circuit that shifts or rotates n bits in one clock cycle

Summary - 2

- Control word specifies the microoperation
- The Instruction Set Architecture (ISA) is the part of the processor that is visible to the programmer or compiler writer including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O
- Instruction decoder is a combinational circuit that provides all control words for the datapath based on the contents of the fields of the instruction
- Single-cycle computer uses 2 distinct memories and has a lower limit on the clock period based on the worst delay path