Register Transfers and Datapaths

Instructor: Saraju P. Mohanty

PART 2

- Datapaths
- Arithmetic and Logic Unit
- The Shifter
- Datapath Representation
- The Control Word
- Pipelined Datapath

Sources

- Logic and Computer Design Fundamentals by M. M. Mano and C. R. Kime.
- Computer Organization and Design: The Hardware/Software Interface by David A. Patterson, John L. Hennessy
- Dr. Valavanis lectures

Datapaths

- Arithmetic / logic unit (ALU) is a shared operation unit that performs various operation on the data from the source registers and stores the result back in the destination register.
- The shift operations may or may not be performed through ALU.
- Datapath: The combination of a set of registers with shared ALU and interconnecting paths.
- CPU or processor: Has two parts the datapath and the controller.
- When large number of registers are included in a datapath, they are most conveniently connected through one or more buses.

Datapath .....

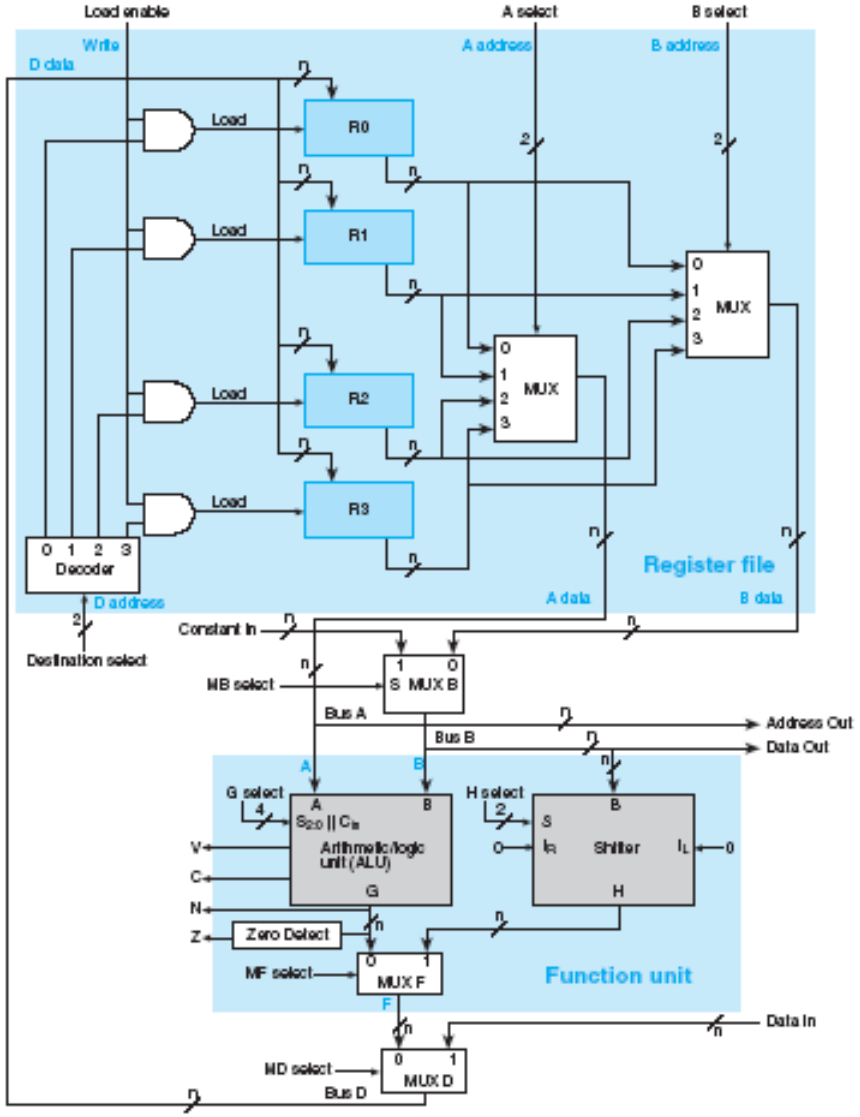


Fig. 7-9 Block Diagram of a Datapath

- There are four registers.
- The registers are connected to two MUXs.
- The select inputs on each multiplexer select one register for the corresponding bus.
- MUX B Multiplexer: used to get constants from outside through bus B.

Datapaths ……

Example: Perform the microoperation, R1 ← R2+R3

1. A select, to place the contents of R2 onto A data (i.e. Bus A)
2. B select, to place the contents of R3 onto the 0 input of MUXB and MB select to put the same data on to Bus B.
3. G select, to provide arithmetic operation A+B.
4. MF select, to place ALU result on the MUXF output.
5. MD select, to put the previous data in Bus D.
6. Destination Select, to select R1 as the destination register.
7. Load enable to load R1 with the above data.

Arithmetic / Logic Unit (ALU)

- ALU is a combinational circuit that performs a set of basic arithmetic and logic microoperations.
- An n-bit ALU combines a n-bit data inputs from A with n data inputs from B to generate the result of an operation at the G outputs.
- Mode select distinguishes between arithmetic and logic operations.
- k-selection lines can specify up to 2^k distinct operations.

Arithmetic / Logic Unit (ALU)

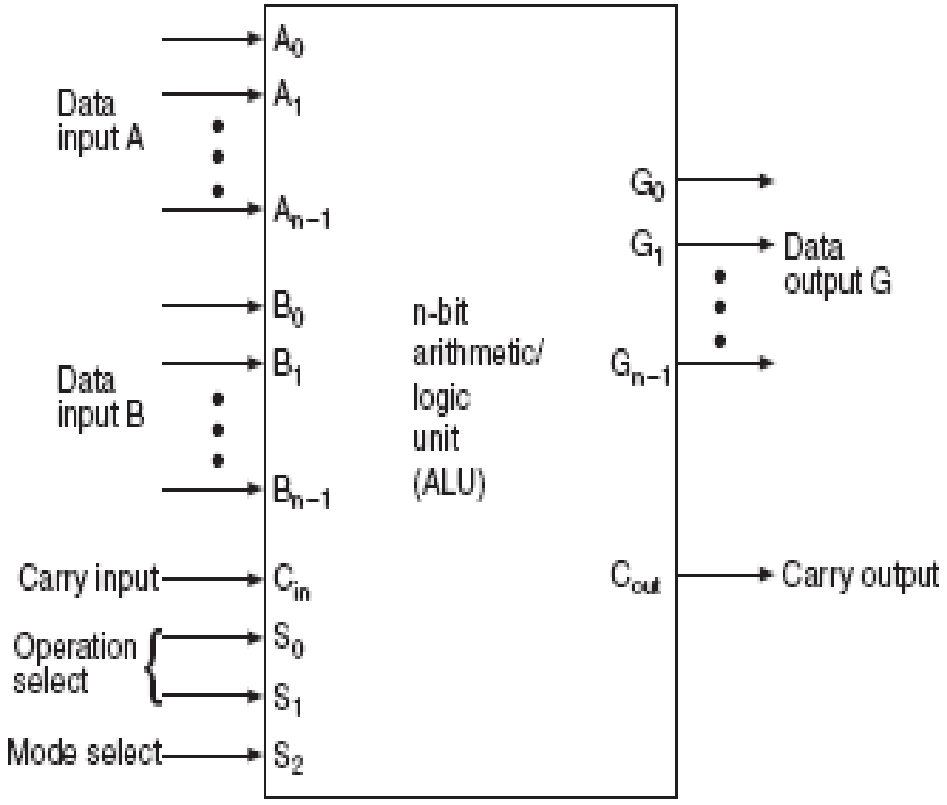


Fig. 7-10 Symbol for an *n*-Bit ALU

- S_2 distinguishes between the arithmetic and logic operations.
- The two function select inputs S_0 and S_1 specify the particular arithmetic and logic operations to be performed.
- Four arithmetic and four logic operations can be selected.
- The Carry input and Carry output have meaning only during an arithmetic operation.

NOTE: ALU design in a three step process as going to be discussed …..

ALU: Arithmetic Unit

Basic component is a parallel adder constructed with a number of full-adder circuits.

One set of inputs to the adder controlled by S_1 and S_0. n inputs from B go through the B input logic to the Y inputs of the parallel adder.

$G = X + Y + C_{in}$

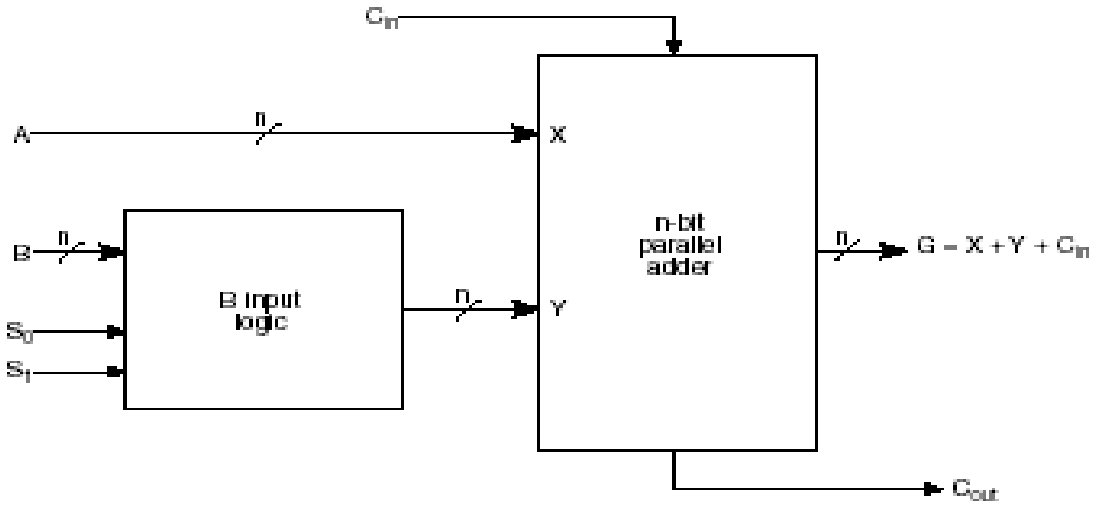


Fig. 7-11 Block Diagram of an Arithmetic Circuit

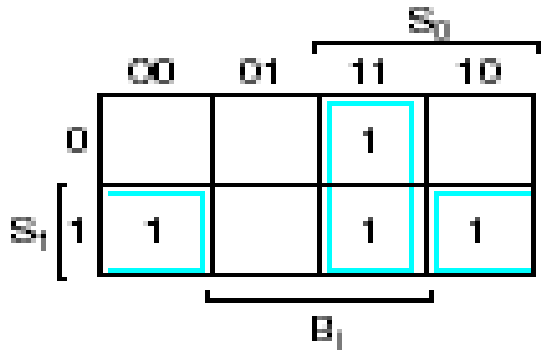
Select		Input	$G = A + Y + C_{in}$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0's	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1's	$G = A - 1$ (decrement)	$G = A$ (transfer)

Table 7-7 Function Table for Arithmetic Circuit

ALU: Arithmetic Unit ….

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	$Y_i = \bar{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

(a) Truth table



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

Fig. 7-12 *B* Input Logic for One Stage of Arithmetic Circuit

ALU: Arithmetic Unit ….

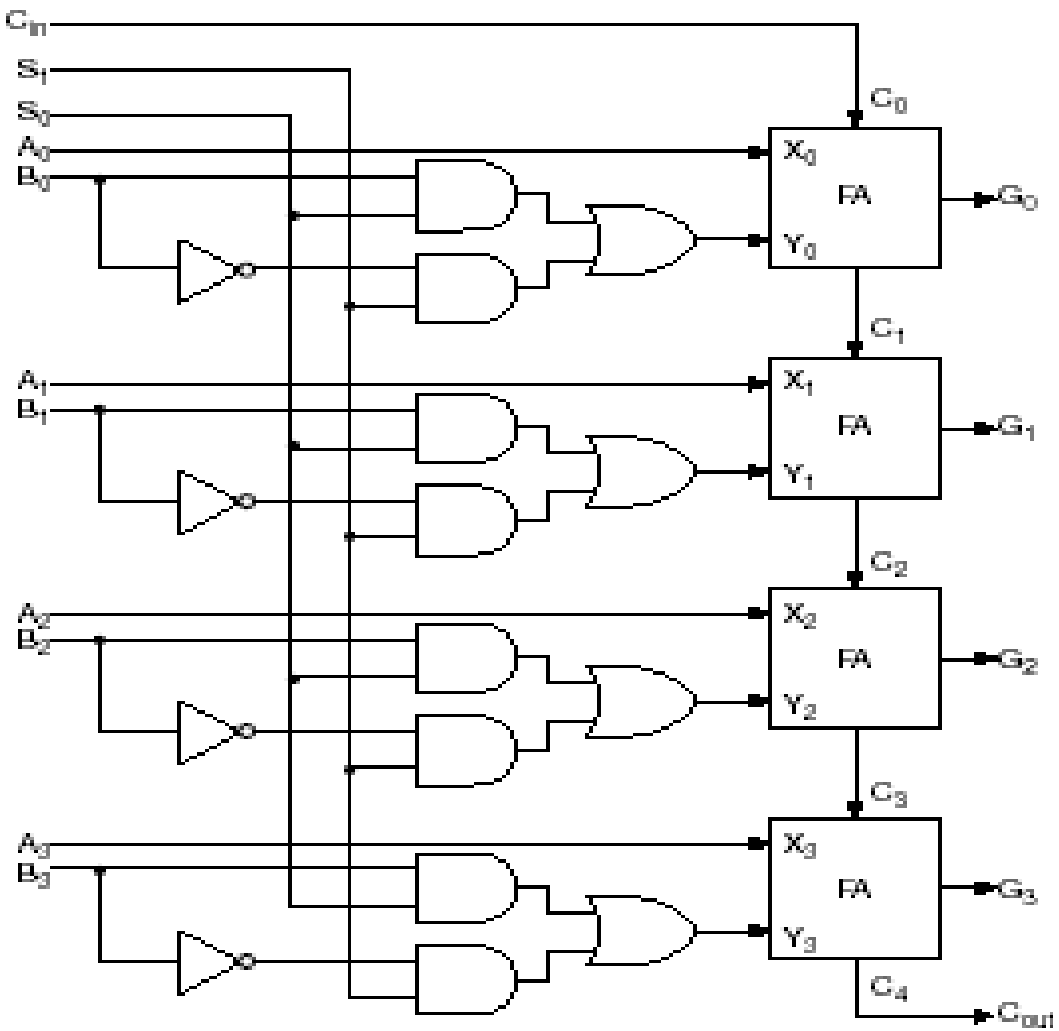
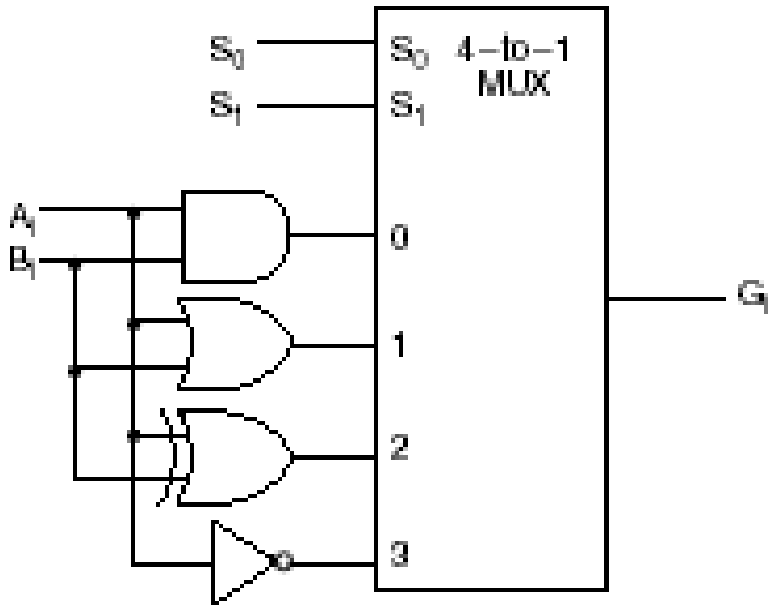


Fig. 7-13 Logic Diagram of a 4-Bit Arithmetic Circuit

- The four full-adder circuits constitute the parallel adder.
- The first stage takes the input carry C_{in}
- All other carries are connected internally from one stage to other.
- The selection variables are S_1, S_0 and C_{in}.

ALU: Logic circuit



(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

Fig. 7-14 One Stage of Logic Circuit

- Four logic operations done bitwise.
- Logic operations are performed and appropriate output is selected.
- More efficient implementations possible.

ALU: Combined, Arithmetic / Logic Unit

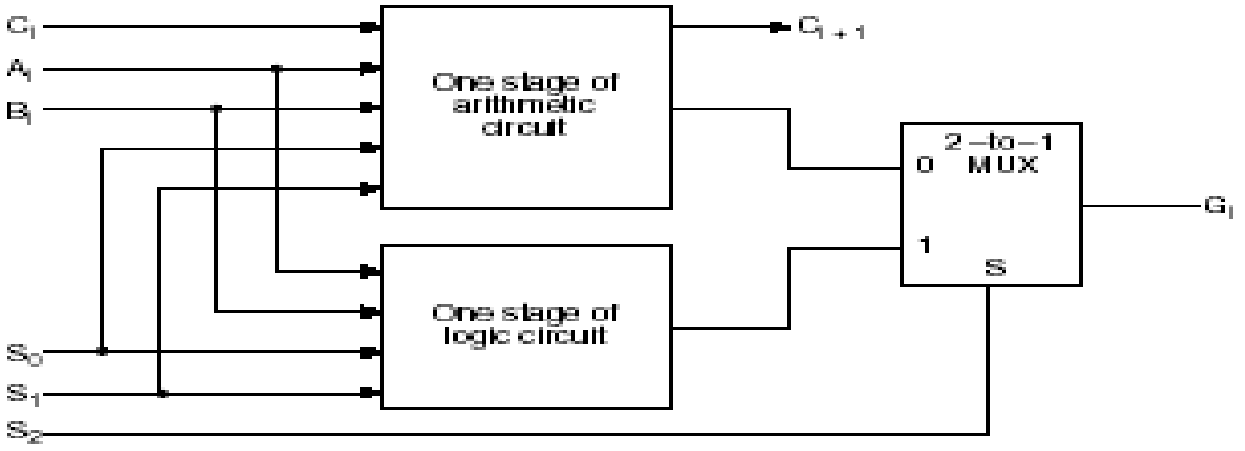


Fig. 7-15 One Stage of ALU

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

Table 7-8 Function Table for ALU

The Shifter

The shifter shifts the value on Bus B, placing the result on an input of MUX F. The basic shifter performs one of two main types of transformations on the data: right and left shift.

Combinatorial shifter: S applied to all 4 MUX to select operation type within the shifter. S=00 causes B to be passed through the shifter unchanged. S=01 causes a right-shift, S=10 a left-shift.

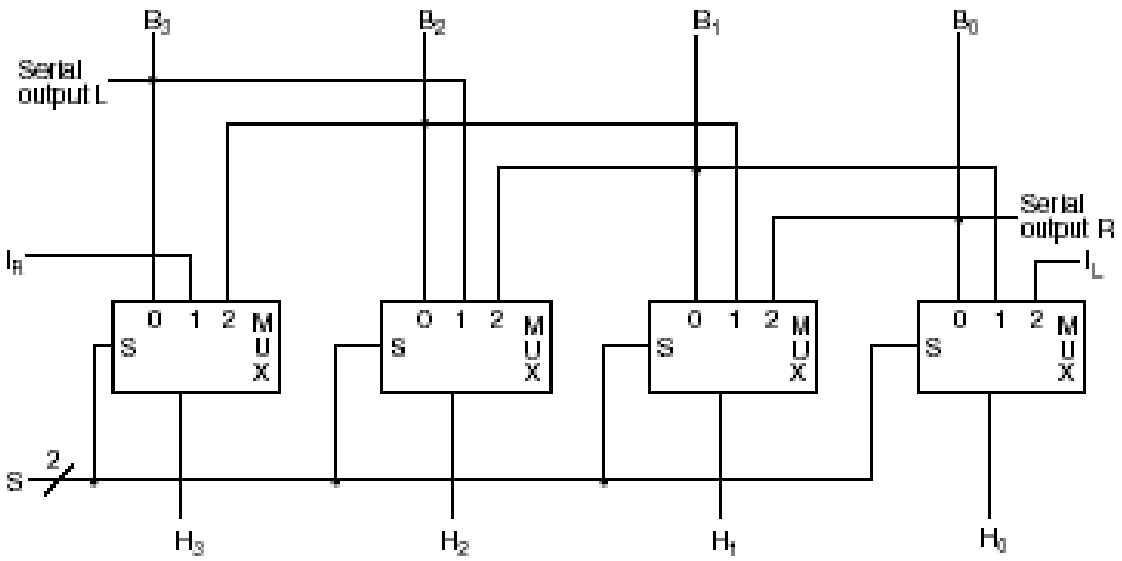


Fig. 7-16 4-Bit Basic Shifter

4 stages of the shifter. In combinatorial shifter, signals propagate through the gates without the need for a clock pulse.

Barrel Shifter

A barrel shifter is one form of combinatorial circuit that shifts or rotates input data bits by the number of bit positions specified by a binary value on a set of selection lines. The 4-bit version is shown below. Common select lines. Selection variables determine the number of positions that the input data will be shifted to the left by rotation. When 00, no shift occurs, input data direct path to output; when 01 input data is rotated one position with D_0 going to Y_1….; when 10, two positions; when 11 three positions.

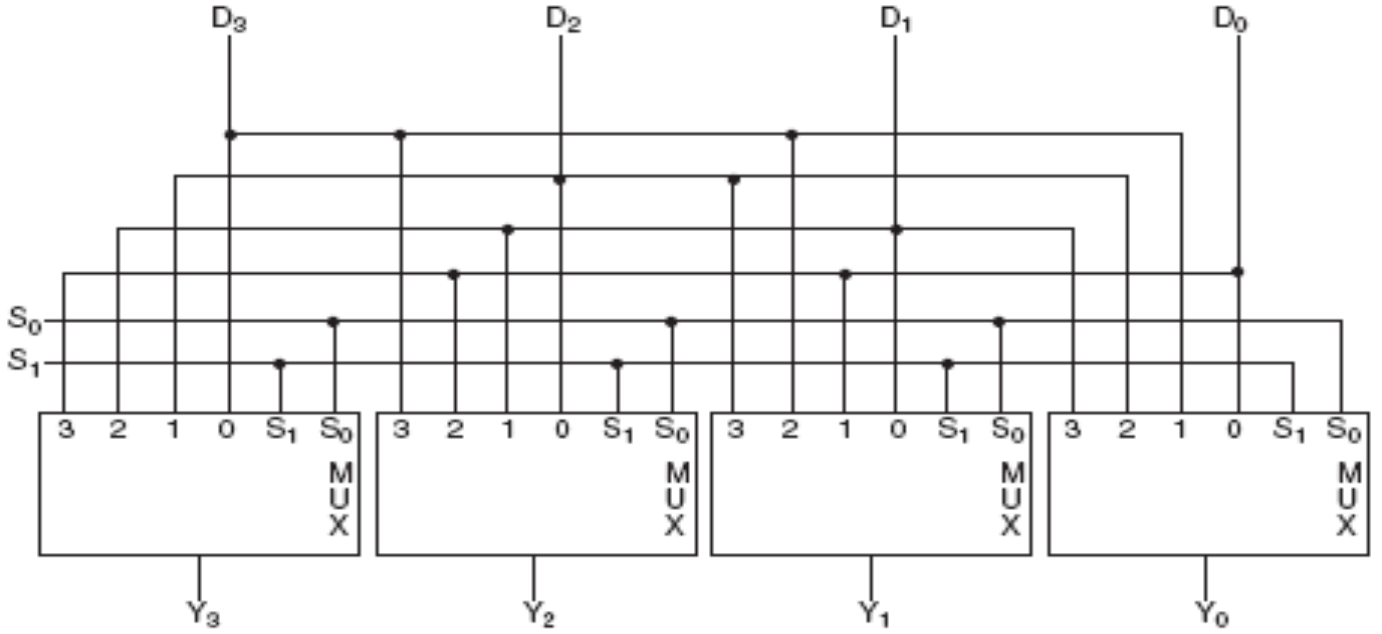


Fig. 7-17 4-Bit Barrel Shifter

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

Table 7-9 Function Table for 4-Bit Barrel Shifter

A barrel shifter with 2^n input and output lines requires 2^n multiplexers, each having 2^n data inputs and n selection inputs. The number of positions for the data to be rotated is specified by the selection variables and can be from 0 to 2^n-1 positions.

Datapath Representation

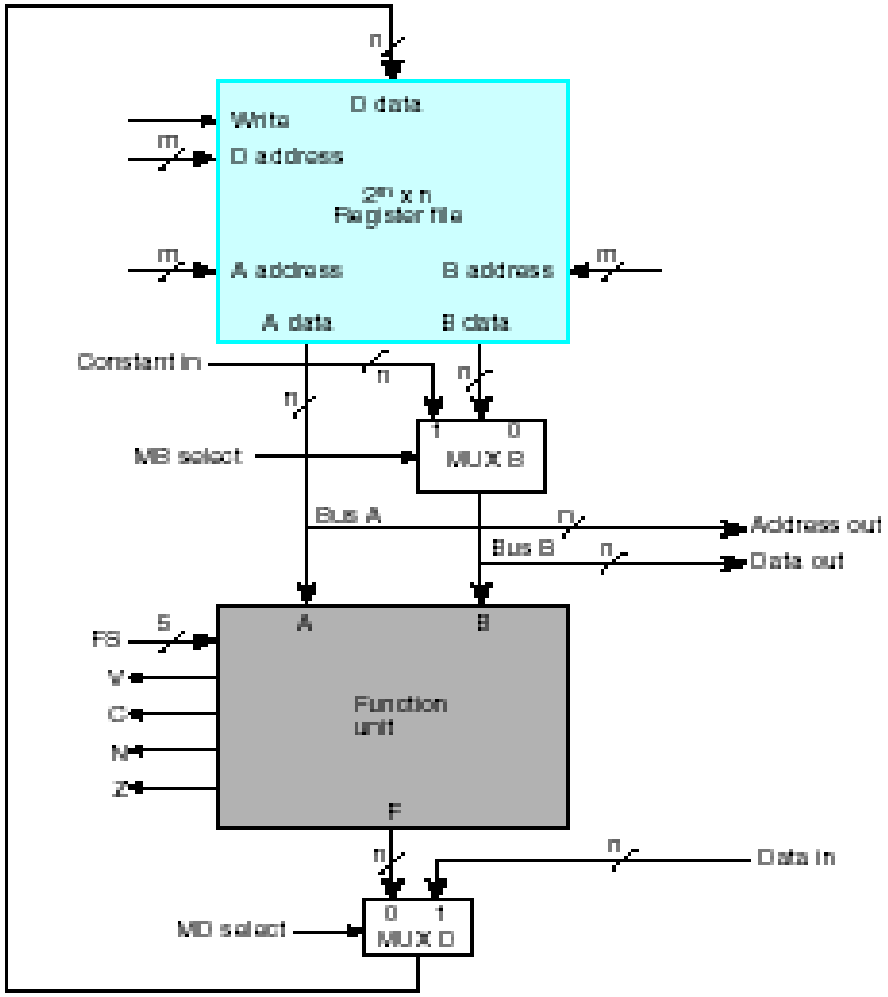


Fig. 7-18 Block Diagram of Datapath Using the Register File and Function Unit

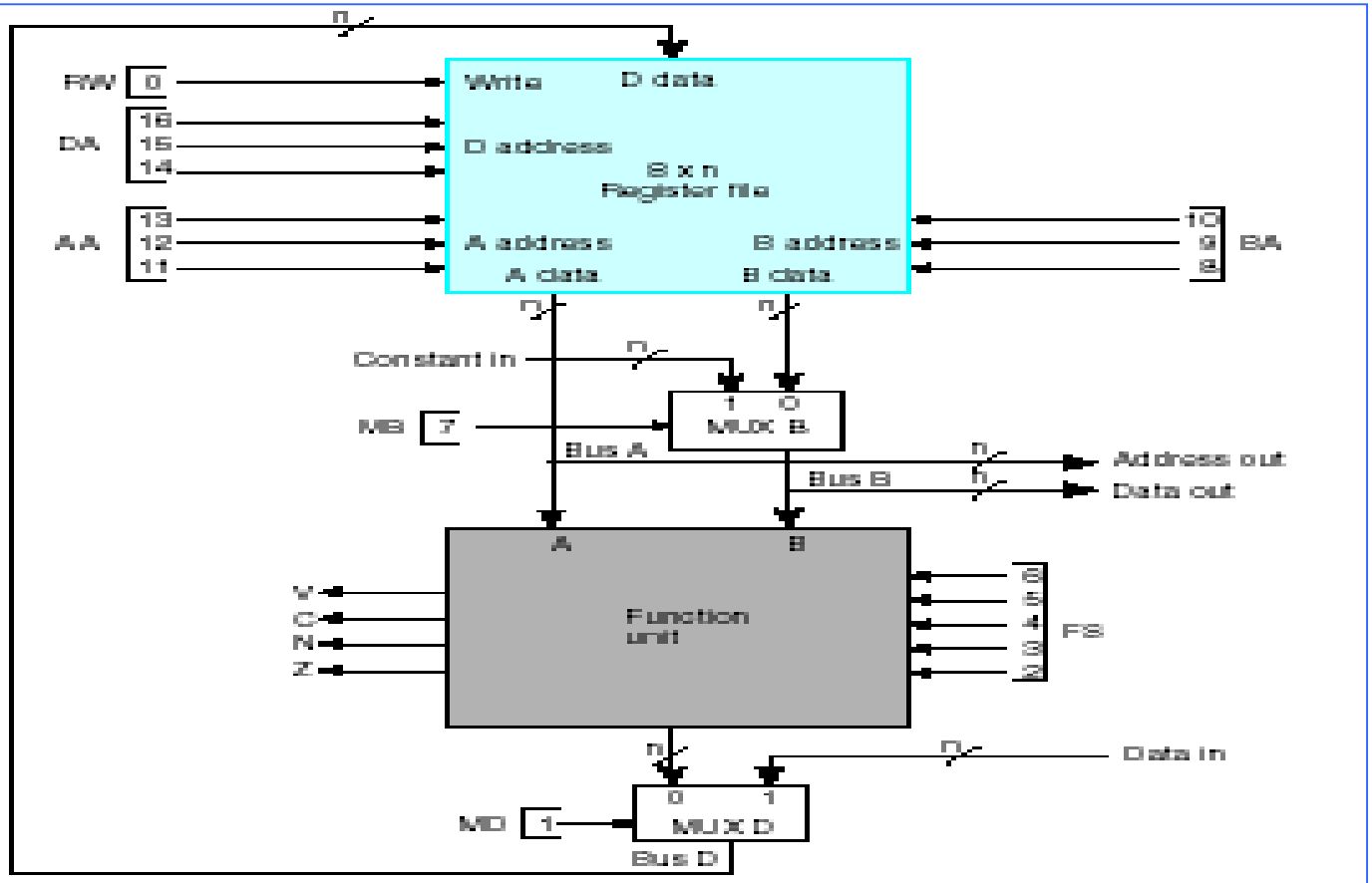
Datapath Representation ….

TABLE 7-10
***G* Select, *H* Select, and *MF* Select Codes Defined in Terms of *FS* Codes**

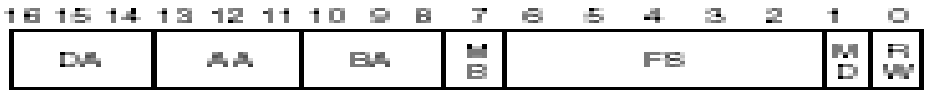
FS	MF Select	G Select	H Select	Microoperation
00000	0	0000	00	$F = A$
00001	0	0001	00	$F = A + 1$
00010	0	0010	00	$F = A + B$
00011	0	0011	00	$F = A + B + 1$
00100	0	0100	01	$F = A + \overline{B}$
00101	0	0101	01	$F = A + B + 1$
00110	0	0110	01	$F = A - 1$
00111	0	0111	01	$F = A$
01000	0	1000	0	$F = A \wedge B$
01010	0	1010	10	$F = A \vee B$
01100	0	1100	10	$F = A \oplus B$
01110	0	1110	10	$F = \overline{A}$
10000	1	0000	00	$F = B$
10100	1	0100	01	$F = \text{sr } B$
11000	1	1000	10	$F = \text{sl } B$

Table 7-10 *G* Select, *H* Select, and *MF* Select Codes Defined in Terms of *FS* Codes

The Control Word



(a) Block Diagram



(b) Control word

Fig. 7-19 Datapath with Control Variables

The Control Word …

TABLE 7-11
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	$F = A$	00000	Function	0	No write	0
$R1$	001	Constant	1	$F = A + 1$	00001	Data In	1	Write	1
$R2$	010			$F = A + B$	00010				
$R3$	011			$F = A + B + 1$	00011				
$R4$	100			$F = A + \overline{B}$ 1	00100				
$R5$	101			$F = A + \overline{B} + 1$	00101				
$R6$	110			$F = A - 1$	00110				
$R7$	111			$F = A$	00111				
				$F = A \wedge B$	01000				
				$F = A \vee B$	01010				
				$F = A \oplus B$	01100				
				$F = \overline{A}$	01110				
				$F = B$	10000				
				$F = \text{sr } B$	10100				
				$F = \text{sl } B$	11000				

Table 7-11 Encoding of Control Word for the Datapath

The Control Word …

TABLE 7-12
Examples of Microoperations for the Datapath, Using Symbolic Notation

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 + \overline{R3} + 1$	$R1$	$R2$	$R3$	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

Table 7-12 Examples of Microoperations for the Datapath, Using Symbolic Notation

The Control Word …

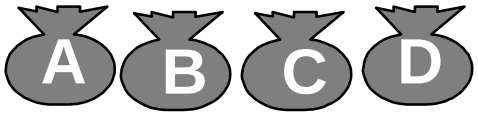
TABLE 7-13
Examples of Microoperations from Table 7-11, Using Binary Control Words

Micro-operation	DA	AA	BA	MB	FS	MD	RW
$R1 \leftarrow R2 - R3$	001	010	011	0	00101	0	1
$R4 \leftarrow$ sl R6	100	000	110	0	11000	0	1
$R7 \leftarrow R7 + 1$	111	111	000	0	00001	0	1
$R1 \leftarrow R0 + 2$	001	000	000	1	00010	0	1
Data out $\leftarrow R3$	000	000	011	0	00000	0	0
$R4 \leftarrow$ Data in	100	000	000	0	00000	1	1
$R5 \leftarrow 0$	101	000	000	0	01100	0	1

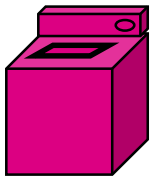
Table 7-13 Examples of Microoperations from Table 7-11, Using Binary Control Words

Pipelining: Laundry Example

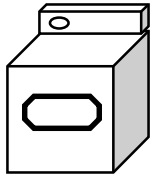
- Ann, Brian, Cathy, Dave each have one load of ciothes to wash, dry, and fold



- Washer takes 30 minutes

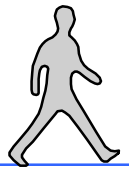


- Dryer takes 30 minutes

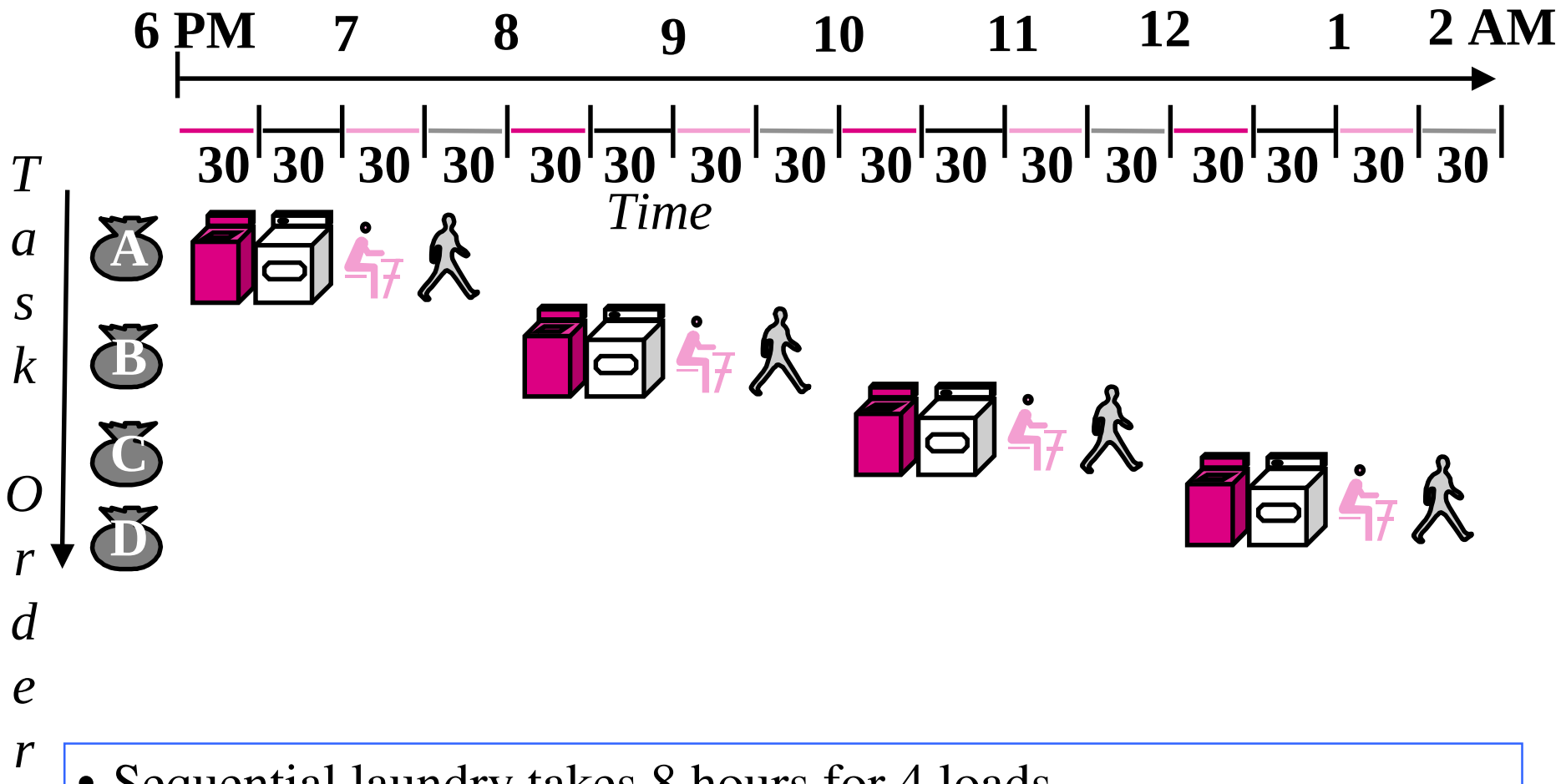


- "Folder" takes 30 minutes

- "Stasher" takes 30 minutes to put clothes into drawers

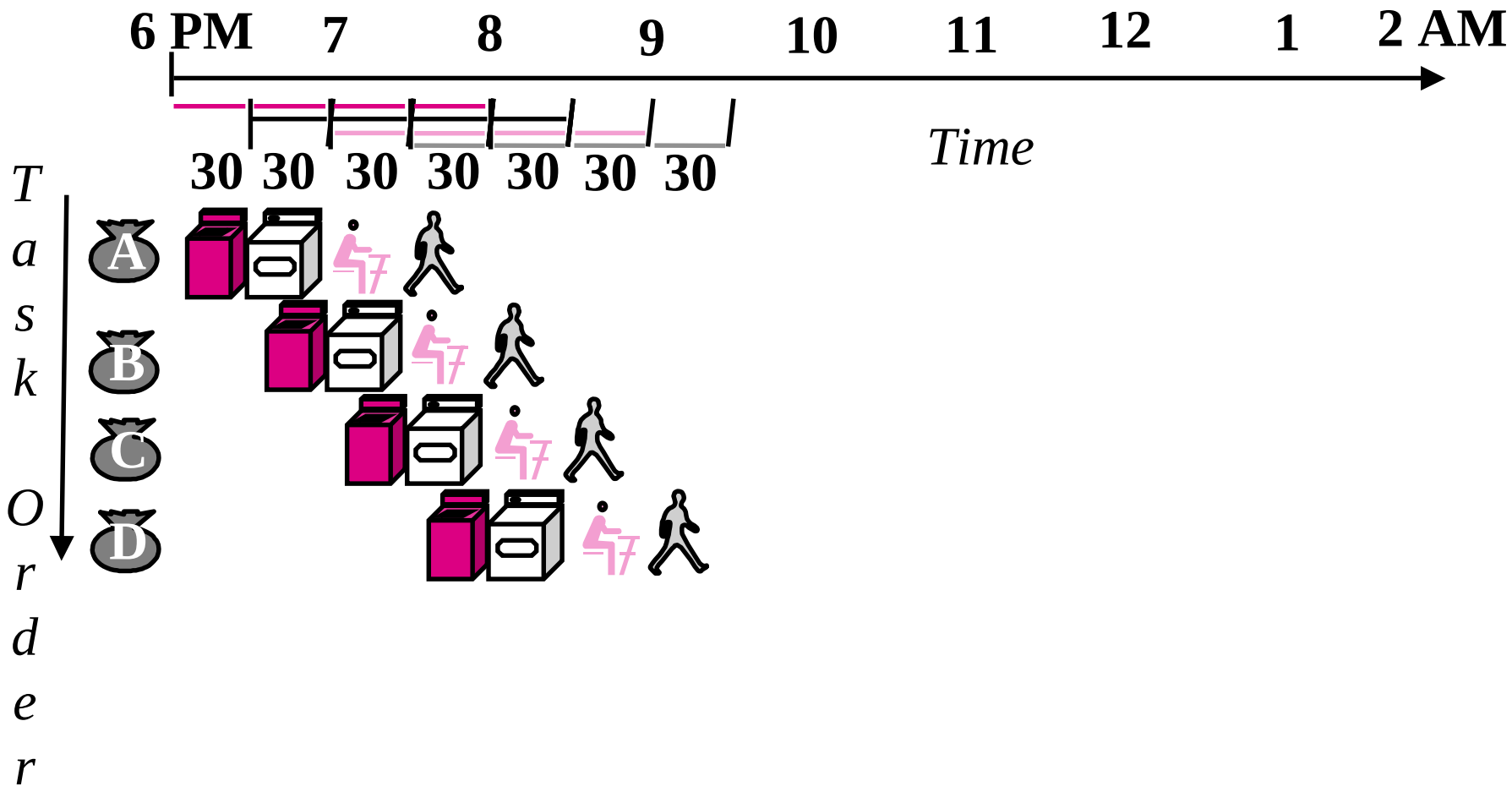


Pipelining: Sequential Laundry



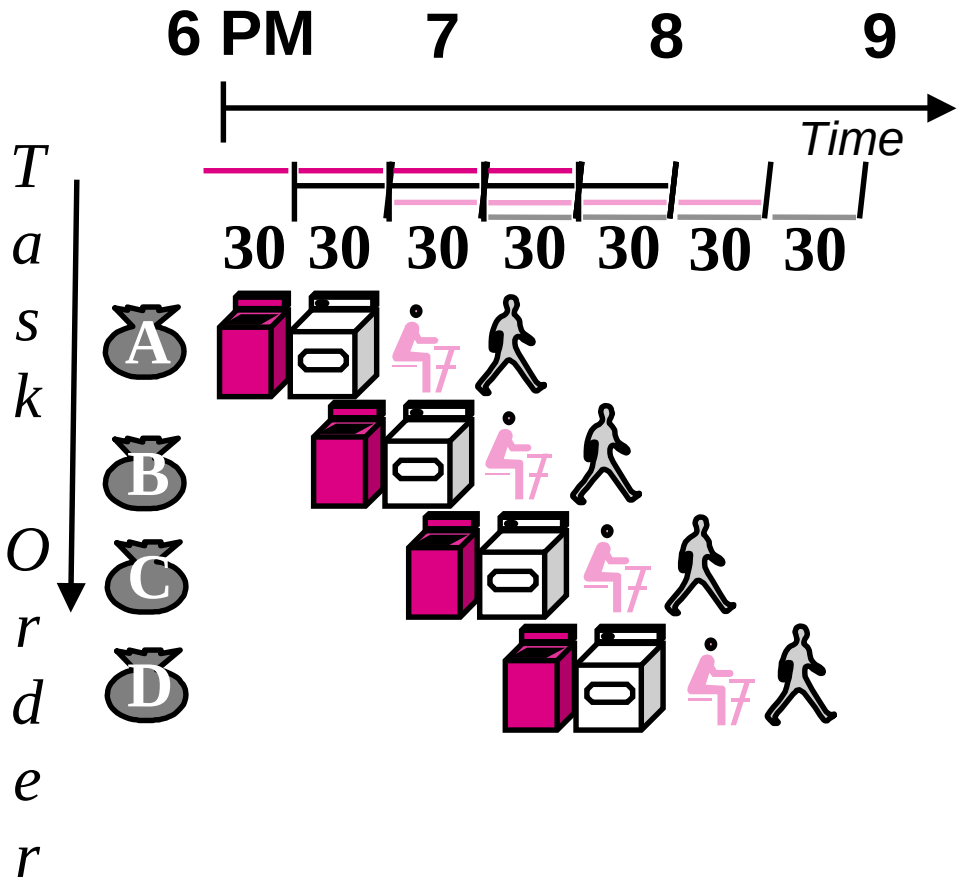
- Sequential laundry takes 8 hours for 4 loads
- If they learned pipelining, how long would laundry take?

Pipelined Laundry: Start work ASAP



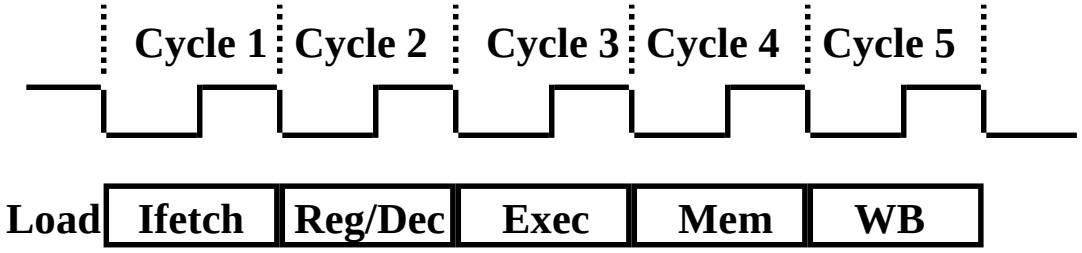
- Pipelined laundry takes 3.5 hours for 4 loads!

Pipelining: Lessons



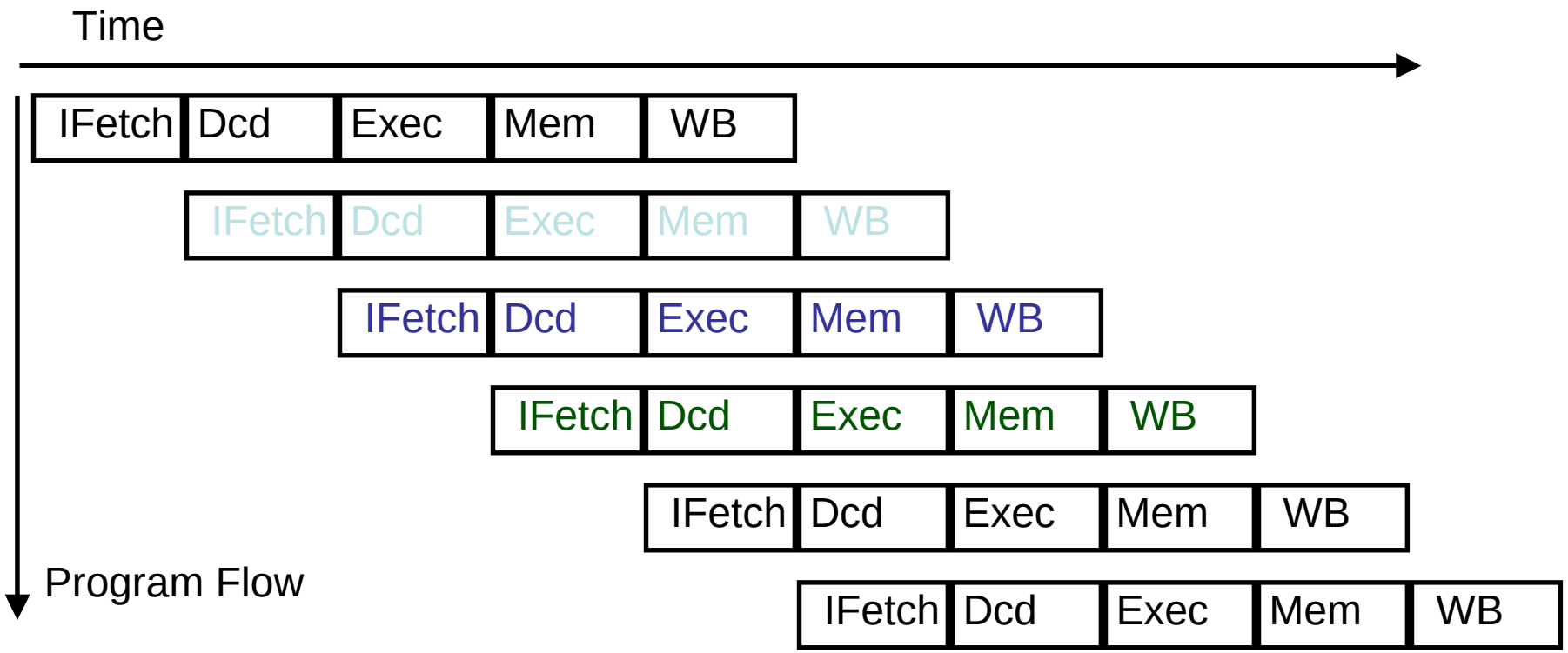
- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Multiple tasks operating simultaneously using different resources
- Potential speedup = Number pipe stages
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup
- Stall for Dependences

Pipelining: The Five Stages of Load

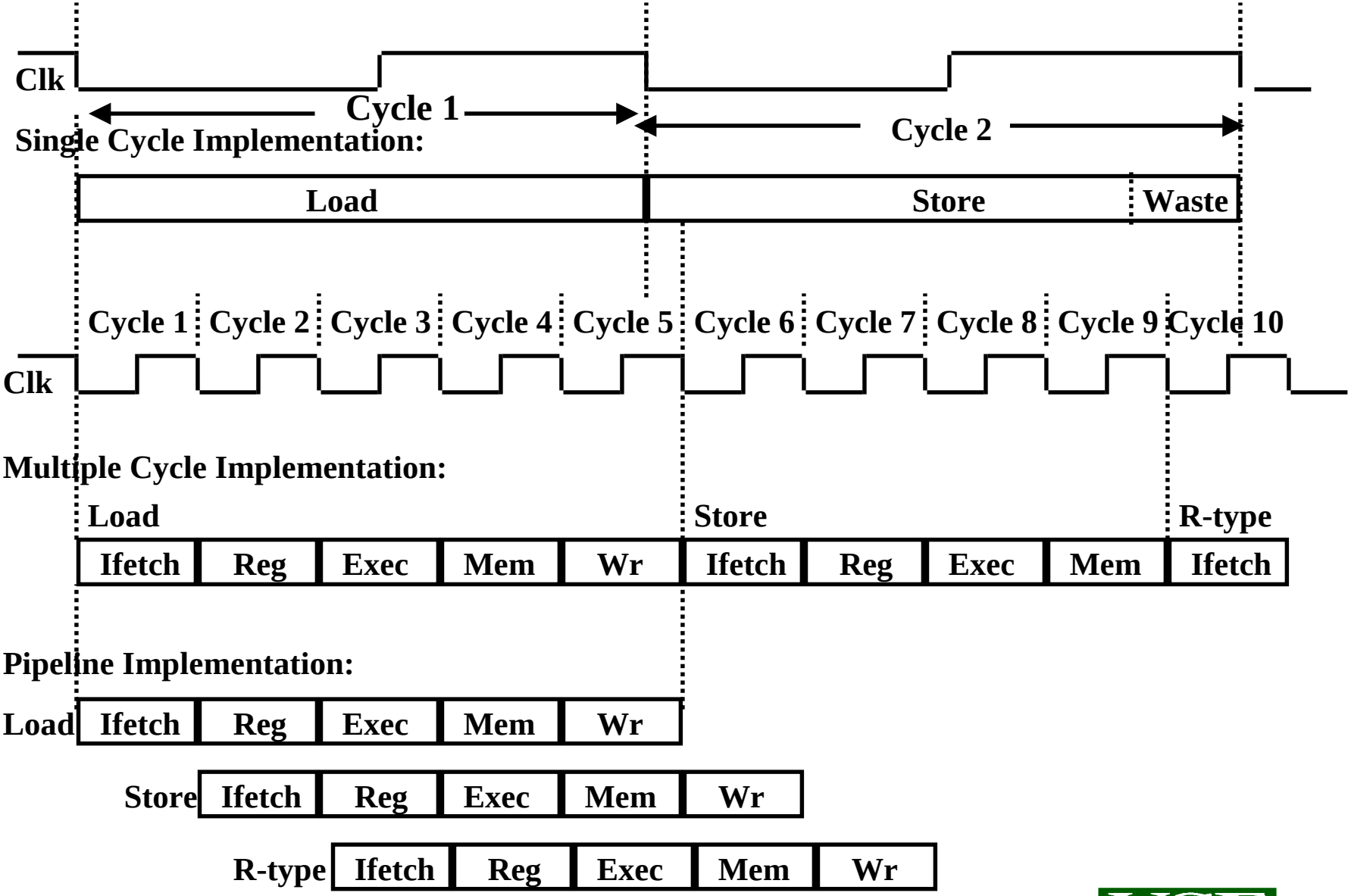


- Ifetch: Instruction Fetch
  - –Fetch the instruction from the Instruction Memory
- Reg/Dec: Registers Fetch and Instruction Decode
- Exec: Calculate the memory address
- Mem: Read the data from the Data Memory
- WB: Write the data back to the register file

Conventional Pipelined Execution Representation



Single Cycle, Multiple Cycle, vs. Pipeline



Pipelined Datapath: 3 stage

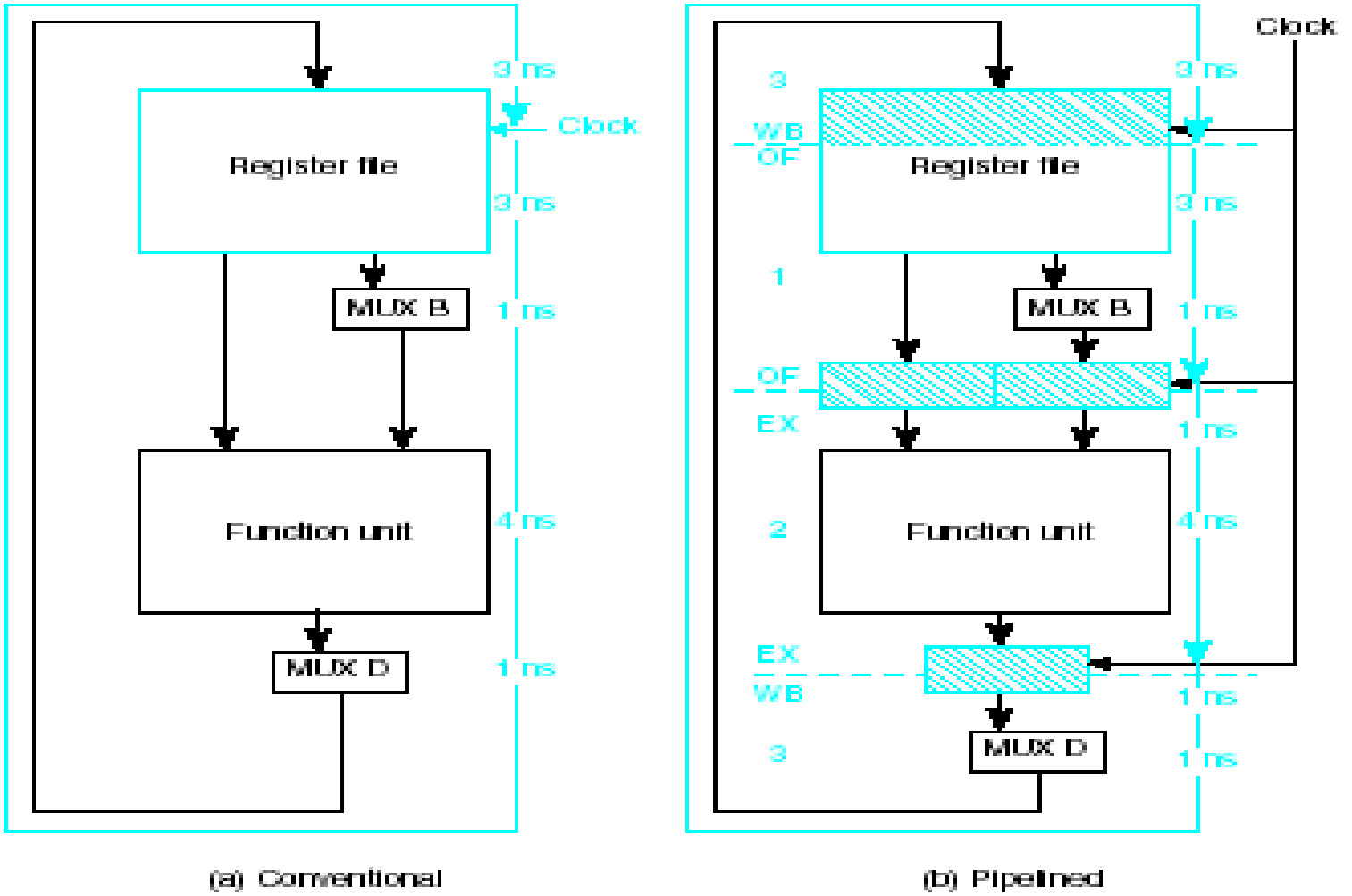


Fig. 7-21 Datapath Timing

Pipelined Datapath: 3 stage ….

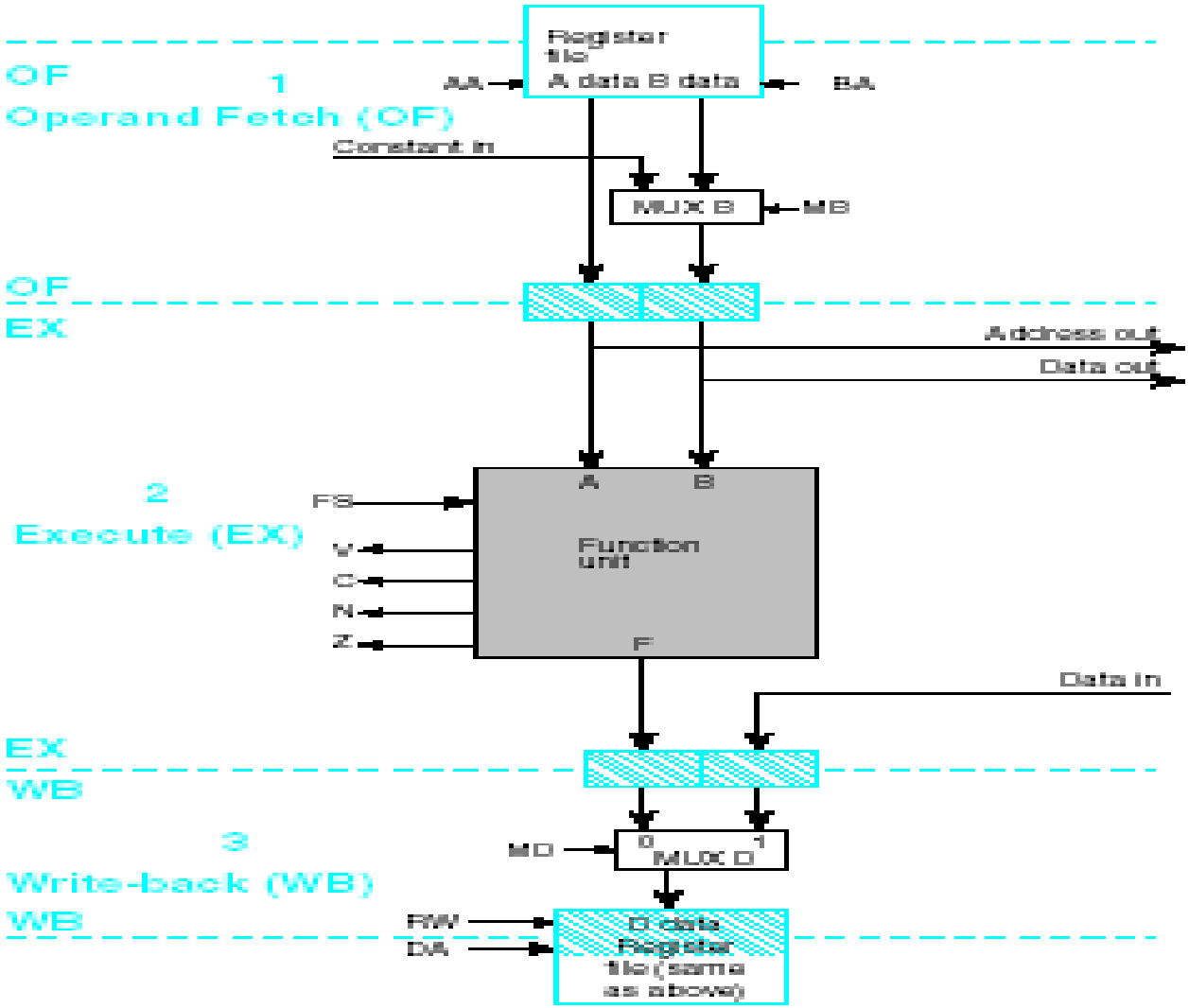


Fig. 7-23 Block Diagram of Pipelined Datapath

Pipelined Datapath: 3 stage Execution

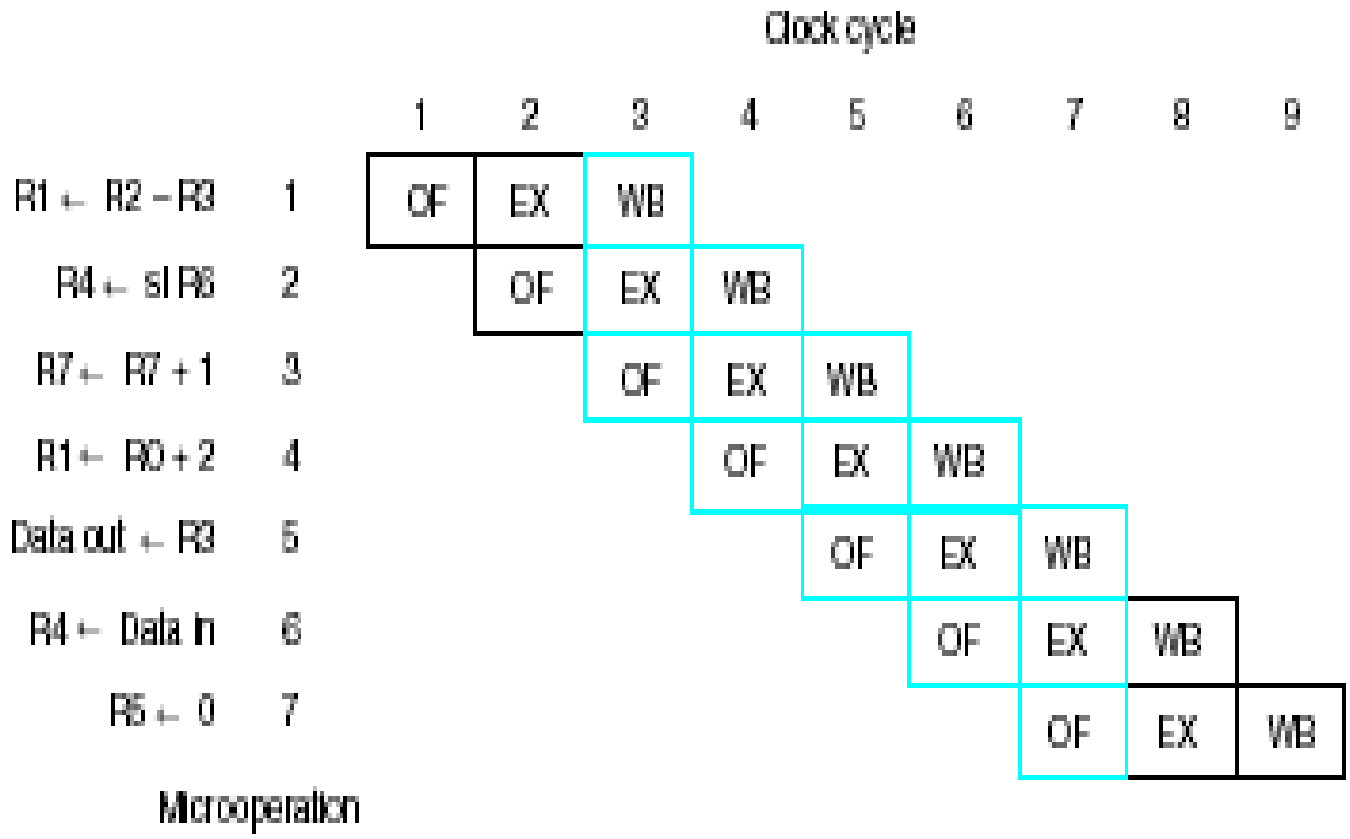


Fig. 7-24 Pipeline Execution Pattern for Microoperation Sequence in Table 7-13