Computer Architecture

Multi-Cycle Hardwired Control Computer

Dr. Masood Mir

Week 10

Multiple- Cycle Hardwired Control

- To demonstrate multiple-cycle control, we use the architecture of the simple single-cycle computer, but modify its **datapath, memory, and control**.
- The block diagram in Figure 8-18 shows the modifications.
- The first modification is to replace the separate instruction memory and data memory with the single Memory M in Figure 8-18.
- To fetch instructions, the **PC** is the address source for the memory, and to fetch data, **Bus A** is the address source. Multiplexer **MUX M** selects between these two address sources. MUX M requires an additional control signal, **MM**, which is added to the control-word.
- Since instructions from Memory M are needed in the control unit, a path is added from its output to the **instruction register IR** in the control unit.

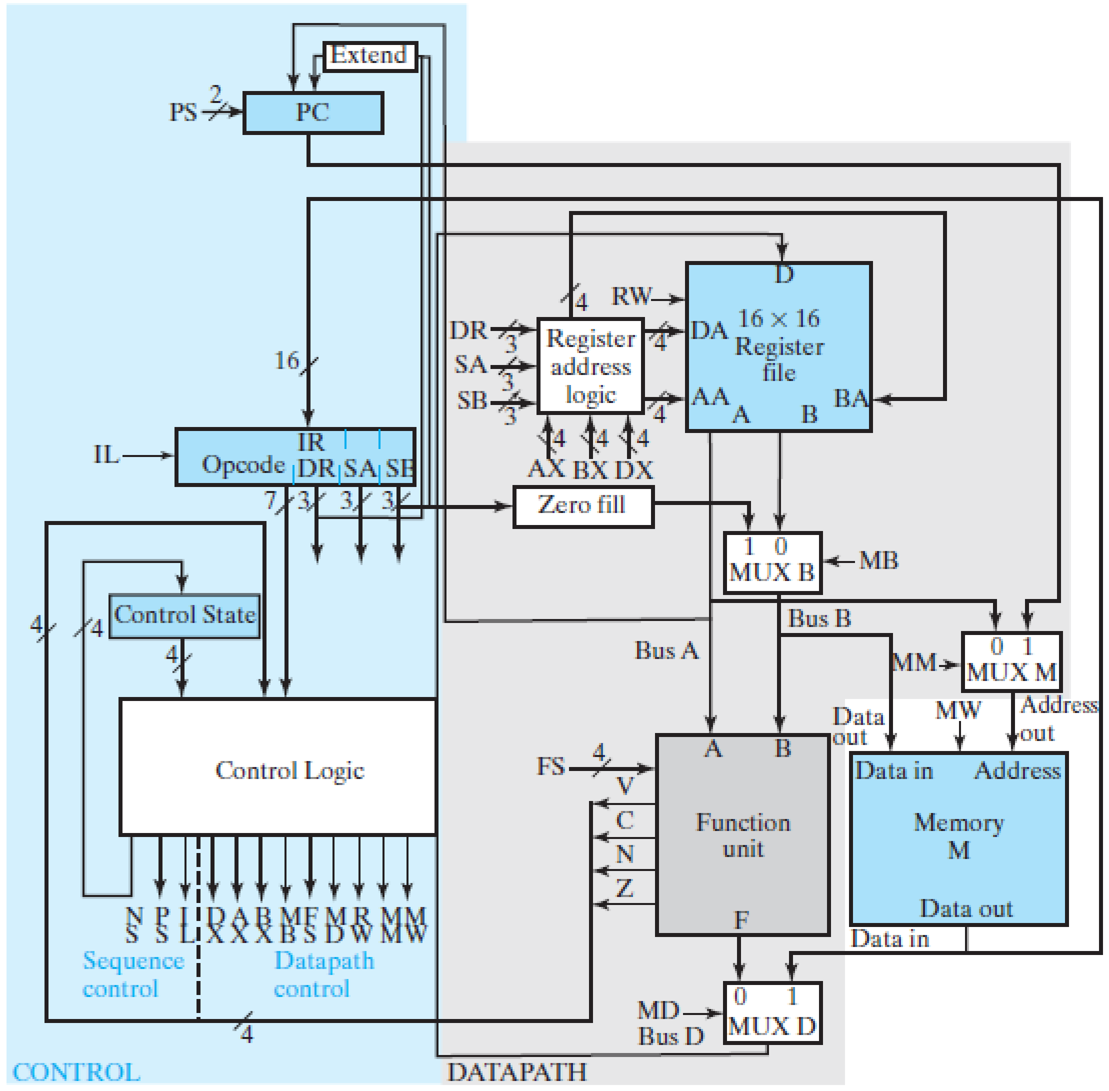


FIGURE 8-18
Block Diagram for a Multiple-Cycle Computer

Multiple- Cycle Hardwired Control vs Single Cyle

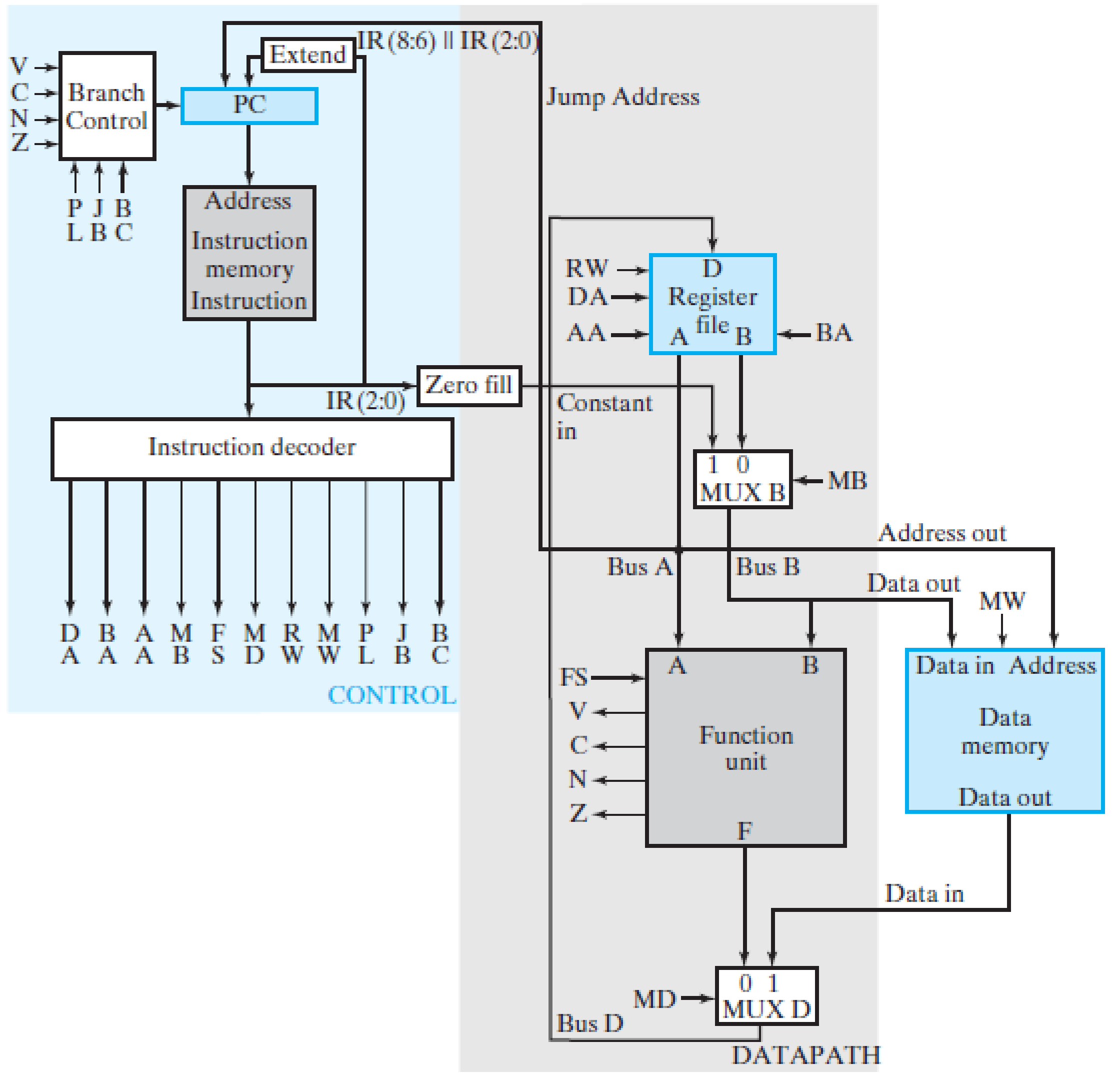


FIGURE 8-15
Block Diagram for a Single-Cycle Computer

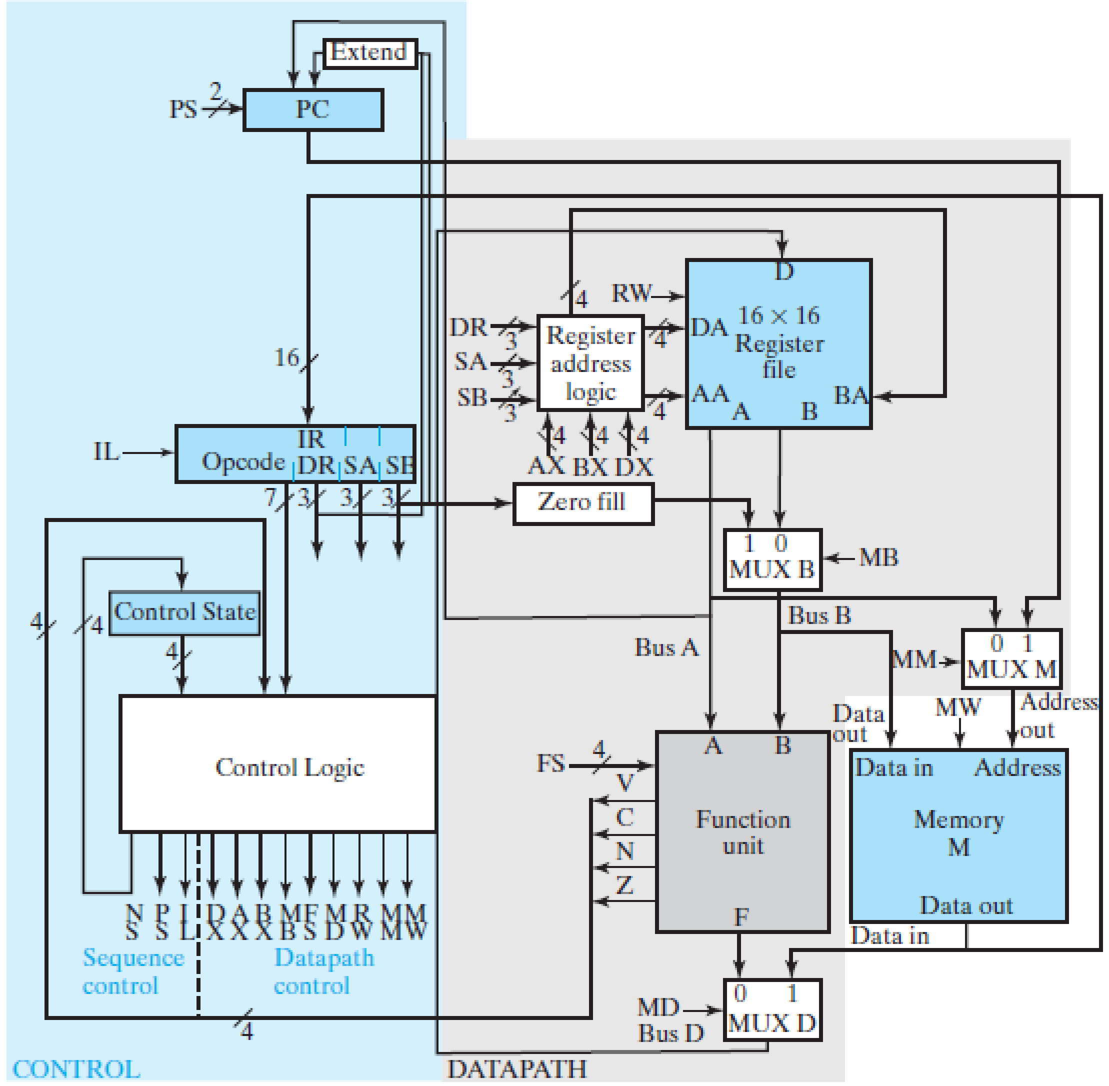


FIGURE 8-18
Block Diagram for a Multiple-Cycle Computer

Multiple- Cycle Hardwired Control

- In executing an instruction across multiple clock cycles, data generated during the current cycle is often needed in a later cycle. This data can be temporarily stored in a register during execution of current instruction. Registers used for such temporary storage during the execution of the instruction are usually not visible to user (i.e., not part of the storage resources).
- The second modification provides these temporary storage registers by **doubling the number of registers** in the register file. **Registers 0 through 7** are storage resources and **Registers 8 through 15** are used only for temporary storage during instruction execution.
- The **addressing of 16 registers requires 4 bits**, and becomes more complex, since addressing of the first eight registers must be controlled from the instruction, and the second eight registers, from the control unit. This is handled by the Register address logic in Figure 8-18 and by modifying **DX, AX, and BX** fields.

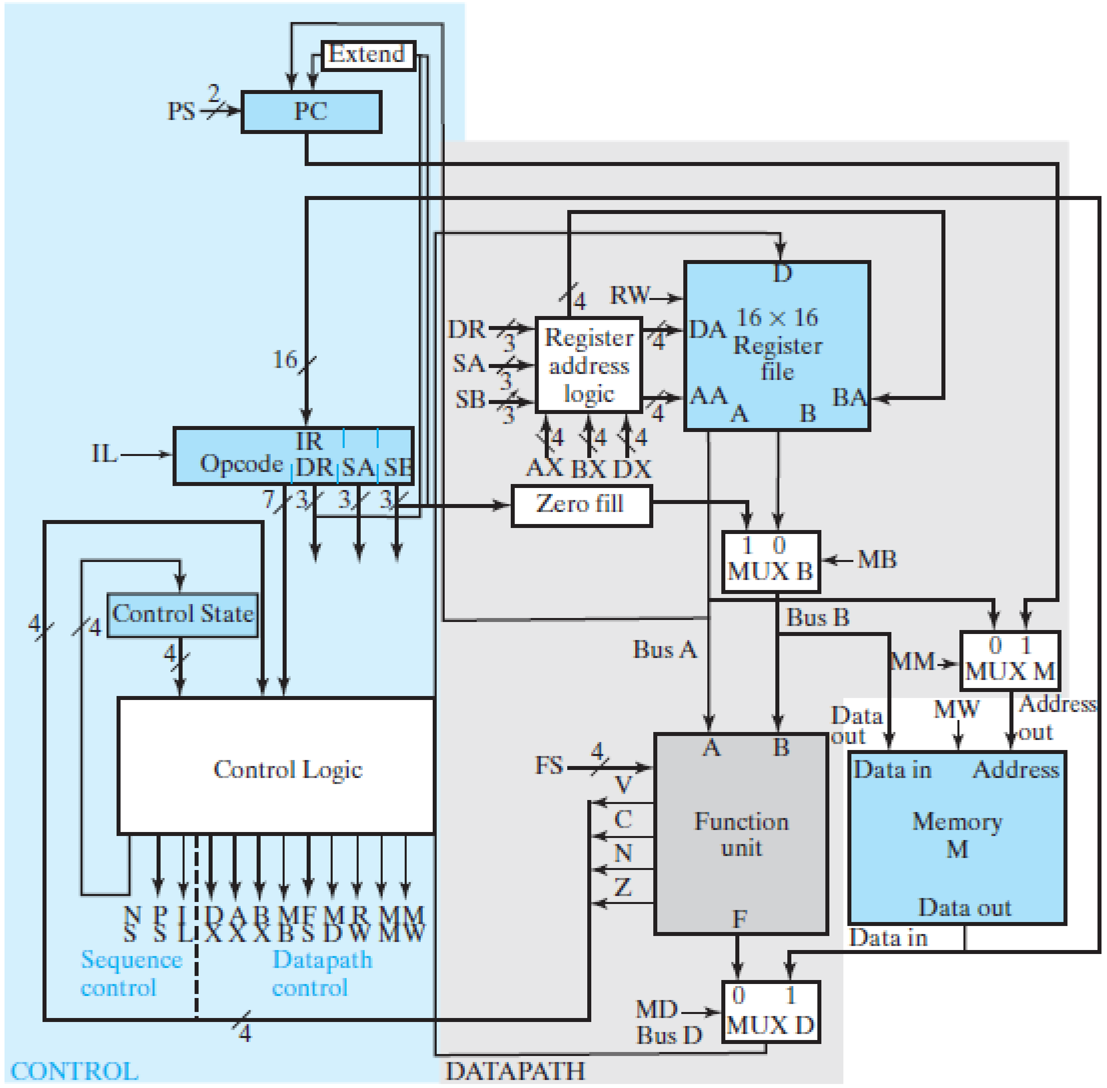


FIGURE 8-18
Block Diagram for a Multiple-Cycle Computer

Multiple- Cycle Hardwired Control

- The PC must also be modified. During the execution of a multiple-cycle instruction, the PC must be held at its current value. To provide this hold capability, as well as an increment and two load operations, the **PC** is modified to be controlled by a 2-bit field, **PS**.
- Since the PC is controlled completely by the control word, the Branch control logic previously represented by **BC** is absorbed into the **Control Logic** block in Figure 8-18 (it is not shown).
- Because of the multiple cycles, the instruction needs to be held in a register for use during its execution since its values are likely to be needed for more than just the first cycle. The register used for this purpose is the **instruction register IR** in Figure 8-18. Since the IR loads only when an instruction is being read from memory, it has a load-enable signal **IL** that is added to the control word.

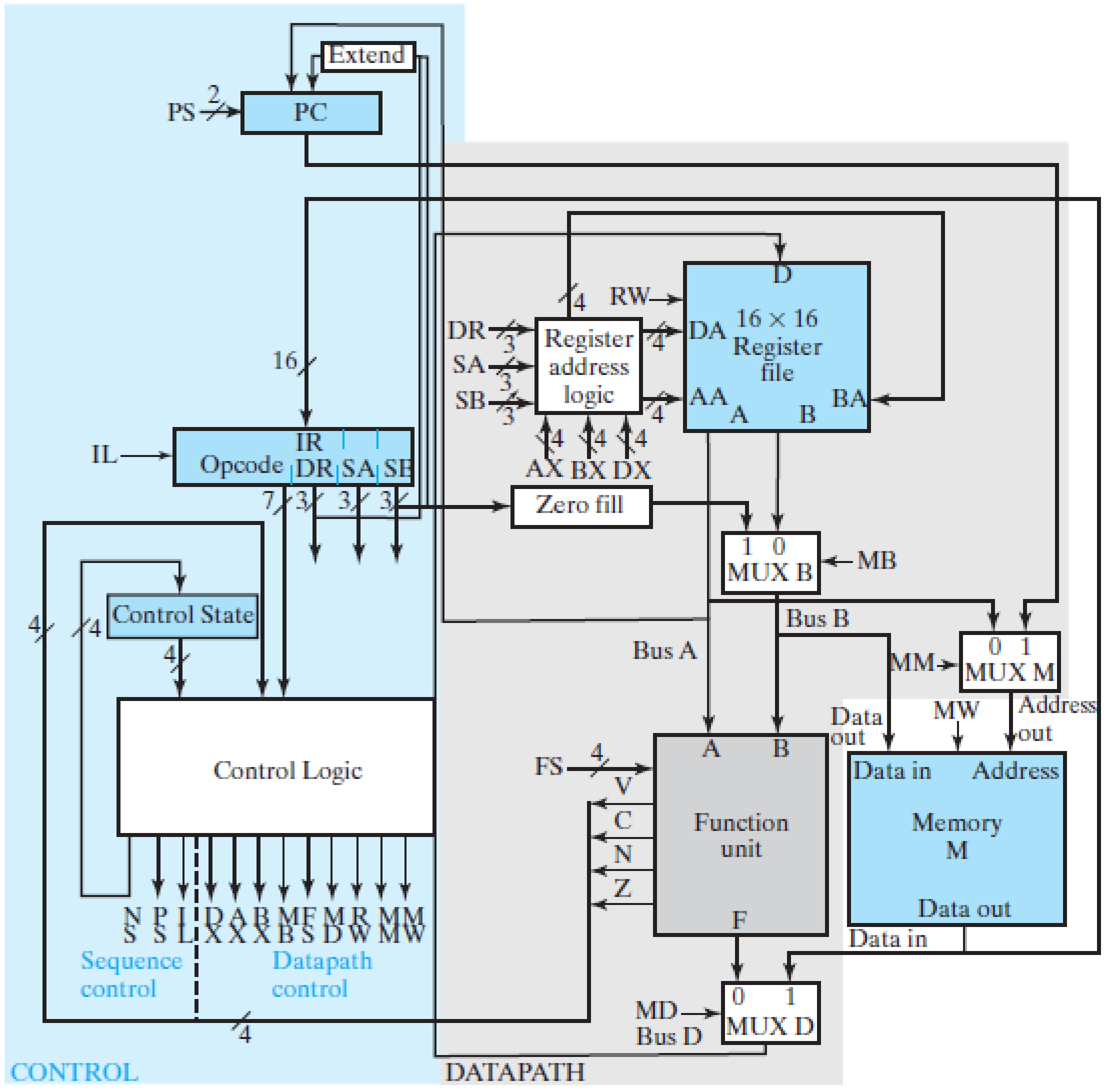


FIGURE 8-18
Block Diagram for a Multiple-Cycle Computer

Multiple- Cycle Hardwired Control

- Because of the multiple-cycle operation, **Instruction Decoder** is replaced by a sequential control circuit, which can provide a sequence of control words for microoperations used to interpret the instruction is required. The sequential control unit consists of the Control state register (**Control State**) and the combinational Control logic. The Control logic has the **state, the opcode, and the status bits** as its inputs and produces the control word as its output.
- Conceptually, the control word is divided into two parts, one for **Sequence control**, which determines the next state of the overall control unit, and one for **Datapath control**, which controls the microoperations executed by the Datapath and Memory M as shown in Figure 8-18.

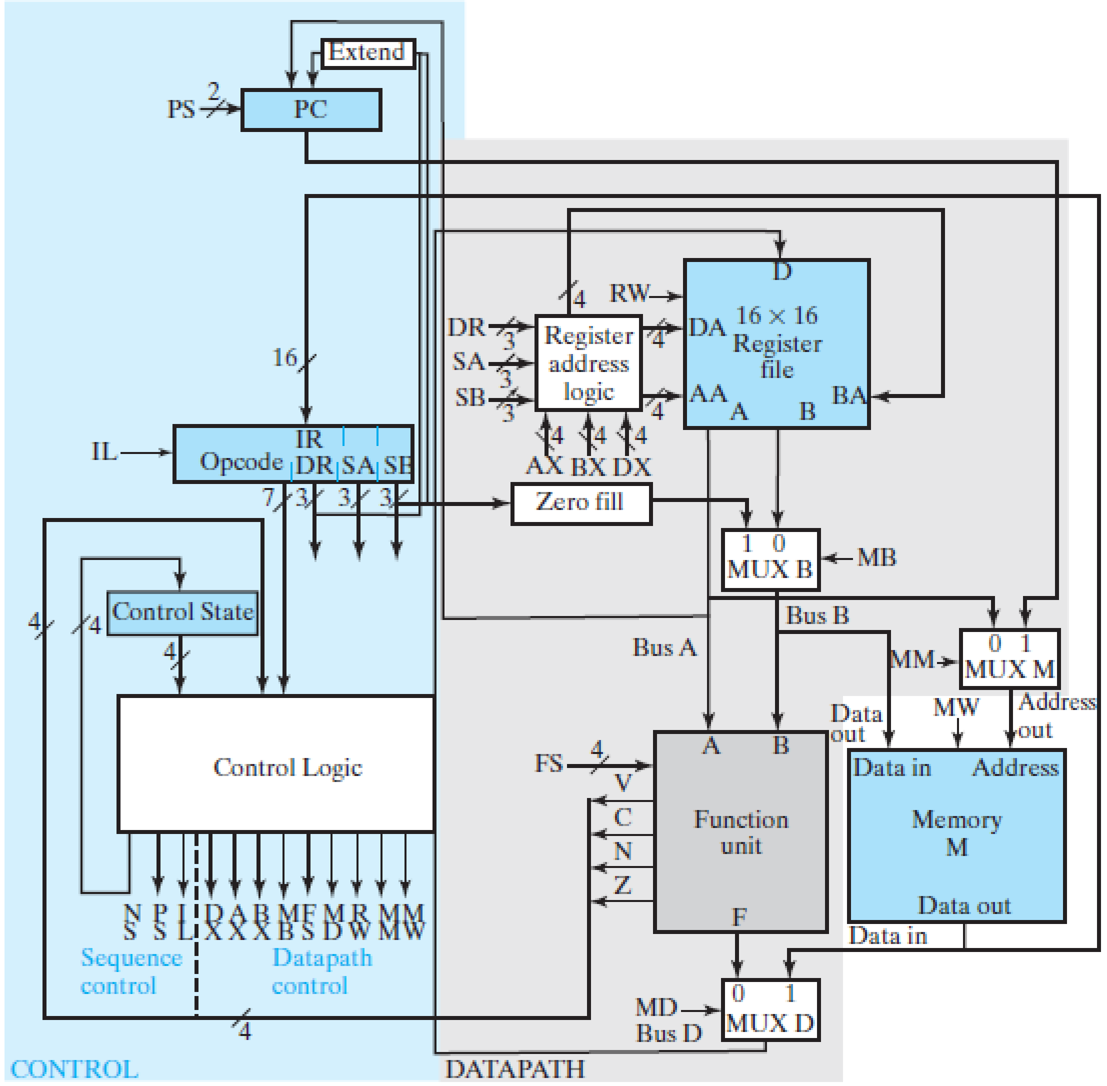


FIGURE 8-18
Block Diagram for a Multiple-Cycle Computer

Multiple- Cycle Hardwired Control

- The **28-bit** modified control word is given in Figure 8-19 and the definitions of the fields of the control word are given in Tables 8-12 and 8-13. In Table 8-12, the fields **DX, AX, and BX** control the register selection.
- If the MSB of one of these fields is 0, then the corresponding registers **DA, AA, or BA** are provided the remaining 3 bits. If the MSB of one of these fields is 1, then the remaining 3 bits register address is for internal registers.
- This selection process is performed by the Register address logic, which contains **three multiplexers**, one for each of DA, AA, and BA, controlled by the MSB of DX, AX, and BX, respectively.
- Table 8-12 also gives the code values for the **MM** field, which determines whether Address out or PC serves as the Memory M address. The remaining fields in Table 8-12, MB, MD, RW, and MW, have the same functions as for the single-cycle computer.

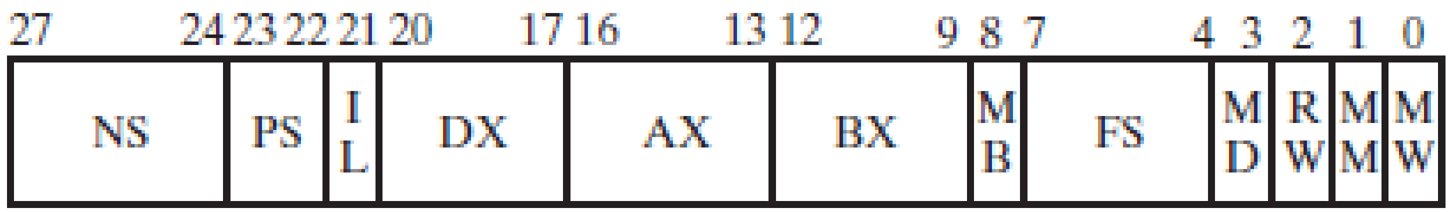


FIGURE 8-19
Control-Word Format for Multiple-Cycle Computer

TABLE 8-12
Control-Word Information for Datapath

DX	AX	BX	Code	MB	Code	FS	Code	MD	RW	MM	MW	Code
R[DR]	*R*[SA]	*R*[SB]	0XXX	Register	0	$F = A$	0000	FnUt	No Write	Address out	No Write	0
R8	*R8*	*R8*	1000	Constant	1	$F = A + 1$	0001	Data in	Write	*PC*	Write	1
R9	*R9*	*R9*	1001			$F = A + B$	0010					
R10	*R10*	*R10*	1010			Unused	0011					
R11	*R11*	*R11*	1011			Unused	0100					
R12	*R12*	*R12*	1100			$F = A+\overline{B}+1$	0101					
R13	*R13*	*R13*	1101			$F = A-1$	0110					
R14	*R14*	*R14*	1110			Unused	0111					
R15	*R15*	*R15*	1111			$F = A \wedge B$	1000					
						$F = A \vee B$	1001					
						$F = A \oplus B$	1010					
						$F = \overline{A}$	1011					
						$F = B$	1100					
						$F = \text{sr } B$	1101					
						$F = \text{sl } B$	1110					
						Unused	1111					

TABLE 8-13
Control Information for Sequence Control

NS	PS		IL	
Next State	**Action**	**Code**	**Action**	**Code**
Gives next state of control state register	Hold PC	00	No load	0
	Inc PC	01	Load IR	1
	Branch	10		
	Jump	11		

Multiple- Cycle Hardwired Control

- In the sequential control unit, the State control register has a set of states, just as any other sequential circuit has. At the level of our discussion, we assume that each state has an abstract name which can be used for present state and the next state value. In the design process, a state assignment needs to be made to these abstract states. Referring to Table 8-13, the field **NS** in the control word provides the next state for the Control State register. We have assigned four bits for the state code, but this can be modified as necessary. The 2-bit **PS** field controls the program counter, PC. On a given clock cycle the PC holds its state (00), increments its state by 1 (01), conditionally loads PC plus sign-extended AD (10), or jumps (unconditionally loads the contents of R[SA] (11)). Finally, the instruction register is loaded only once during the execution of an instruction. Thus, on any given cycle, either a new instruction is loaded (IL = 1) or the instruction remains unchanged (IL = 0).

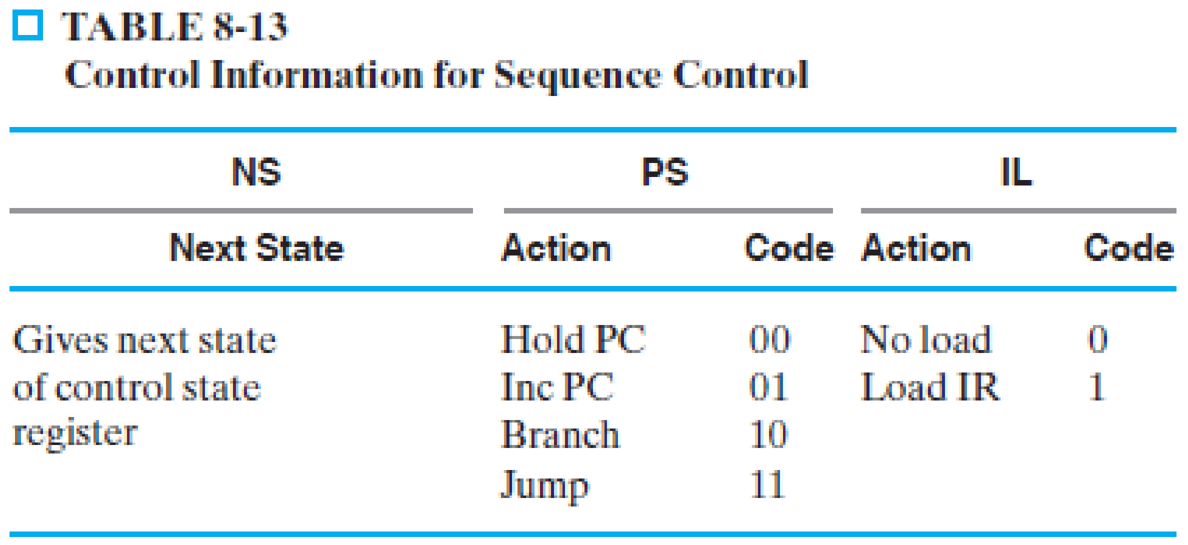
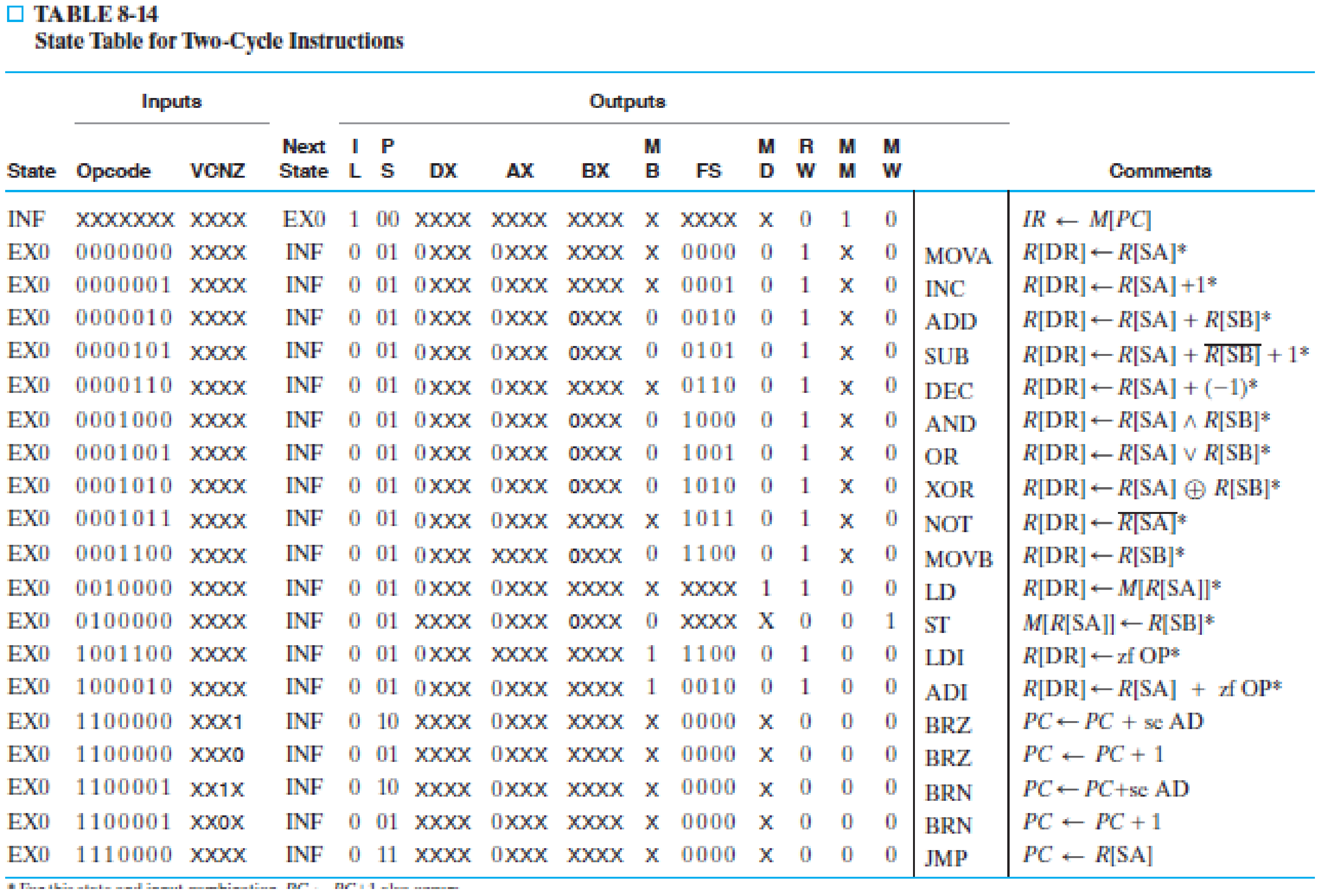


TABLE 8-13
Control Information for Sequence Control

NS	PS		IL	
Next State	Action	Code	Action	Code
Gives next state of control state register	Hold PC	00	No load	0
	Inc PC	01	Load IR	1
	Branch	10		
	Jump	11		

Multiple- Cycle Hardwired Control

- It is convenient to separate the cycles into two processing steps: instruction fetch and instruction execution. **Instruction fetch** occurs in state **INF** at the top of the chart. The PC contains the address of the instruction in Memory M. This address is applied to the memory, and the word read from memory is loaded into the IR on the positive clock edge that ends state INF.



☐ **TABLE 8-14**
State Table for Two-Cycle Instructions

	Inputs			Outputs													
State	Opcode	VCNZ	Next State	IL	PS	DX	AX	BX	MB	FS	MD	RW	MM	MW		Comments
INF	xxxxxxx	xxxx	EX0	1	00	xxxx	xxxx	xxxx	x	xxxx	x	0	1	0		$IR \leftarrow M[PC]$
EX0	0000000	xxxx	INF	0	01	0xxx	0xxx	xxxx	x	0000	0	1	x	0	MOVA	$R[\text{DR}] \leftarrow R[\text{SA}]$*
EX0	0000001	xxxx	INF	0	01	0xxx	0xxx	xxxx	x	0001	0	1	x	0	INC	$R[\text{DR}] \leftarrow R[\text{SA}]+1$*
EX0	0000010	xxxx	INF	0	01	0xxx	0xxx	0xxx	0	0010	0	1	x	0	ADD	$R[\text{DR}] \leftarrow R[\text{SA}] + R[\text{SB}]$*
EX0	0000101	xxxx	INF	0	01	0xxx	0xxx	0xxx	0	0101	0	1	x	0	SUB	$R[\text{DR}] \leftarrow R[\text{SA}] + \overline{R[\text{SB}]} + 1$*
EX0	0000110	xxxx	INF	0	01	0xxx	0xxx	xxxx	x	0110	0	1	x	0	DEC	$R[\text{DR}] \leftarrow R[\text{SA}] + (-1)$*
EX0	0001000	xxxx	INF	0	01	0xxx	0xxx	0xxx	0	1000	0	1	x	0	AND	$R[\text{DR}] \leftarrow R[\text{SA}] \wedge R[\text{SB}]$*
EX0	0001001	xxxx	INF	0	01	0xxx	0xxx	0xxx	0	1001	0	1	x	0	OR	$R[\text{DR}] \leftarrow R[\text{SA}] \vee R[\text{SB}]$*
EX0	0001010	xxxx	INF	0	01	0xxx	0xxx	0xxx	0	1010	0	1	x	0	XOR	$R[\text{DR}] \leftarrow R[\text{SA}] \oplus R[\text{SB}]$*
EX0	0001011	xxxx	INF	0	01	0xxx	0xxx	xxxx	x	1011	0	1	x	0	NOT	$R[\text{DR}] \leftarrow \overline{R[\text{SA}]}$*
EX0	0001100	xxxx	INF	0	01	0xxx	xxxx	0xxx	0	1100	0	1	x	0	MOVB	$R[\text{DR}] \leftarrow R[\text{SB}]$*
EX0	0010000	xxxx	INF	0	01	0xxx	0xxx	xxxx	x	xxxx	1	1	0	0	LD	$R[\text{DR}] \leftarrow M[R[\text{SA}]]$*
EX0	0100000	xxxx	INF	0	01	xxxx	0xxx	0xxx	0	xxxx	X	0	0	1	ST	$M[R[\text{SA}]] \leftarrow R[\text{SB}]$*
EX0	1001100	xxxx	INF	0	01	0xxx	xxxx	xxxx	1	1100	0	1	0	0	LDI	$R[\text{DR}] \leftarrow$ zf OP*
EX0	1000010	xxxx	INF	0	01	0xxx	0xxx	xxxx	1	0010	0	1	0	0	ADI	$R[\text{DR}] \leftarrow R[\text{SA}] +$ zf OP*
EX0	1100000	xxx1	INF	0	10	xxxx	0xxx	xxxx	x	0000	x	0	0	0	BRZ	$PC \leftarrow PC +$ se AD
EX0	1100000	xxx0	INF	0	01	xxxx	0xxx	xxxx	x	0000	x	0	0	0	BRZ	$PC \leftarrow PC + 1$
EX0	1100001	xx1x	INF	0	10	xxxx	0xxx	xxxx	x	0000	x	0	0	0	BRN	$PC \leftarrow PC +$ se AD
EX0	1100001	xx0x	INF	0	01	xxxx	0xxx	xxxx	x	0000	x	0	0	0	BRN	$PC \leftarrow PC + 1$
EX0	1110000	xxxx	INF	0	11	xxxx	0xxx	xxxx	x	0000	x	0	0	0	JMP	$PC \leftarrow R[\text{SA}]$

* For this state and input combination, $PC \leftarrow PC+1$ also occurs

Multiple- Cycle Hardwired Control

- The same clock edge causes the new state to become EX0. In state EX0, the instruction is decoded and the microoperations executing all or part of the instruction appear in Mealy-type outputs.
- If the instruction can be completed in state EX0, the next state is INF in preparation for fetching of the next instruction. Further, for instructions that do not change PC contents during their execution, the PC is incremented.
- If additional states are required for instruction execution, the next state is EX1. In each of the execution states, 128 different input combinations are possible, based on the opcode.
- Many of these opcodes will be unused. We assume that these opcodes will never appear and so will be don’t-care inputs.
- An alternative assumption is that if they do appear, they cause an exception that signals their presence.

Multiple- Cycle New Instructions Examples

- Load Register Indirect” (LRI)

  In this instruction, the contents of register SA address a word in memory. The word, which is known as an indirect address, is then used to address the word in memory that is loaded into register DR. This can be represented symbolically as R[DR] <- M[M[R[SA]]]
- Shift Right Multiple (SRM) and Shift Left Multiple (SLM)

  The final two instructions to be added are “shift right multiple” (SRM) and “shift left multiple” (SLM), with opcodes 0001101 and 0001110, respectively. SRM specifies that the contents of register SA are to be shifted to the right by the number of positions given by the three bits of the OP field, with the result placed in register DR. And SLM does the same in left direction.
- The point that you need to learn from these multiple cycle instructions is that, first an instruction is fetched (INF) and than executed (going to EX0). If EX0 do not complete the execution, next state will be EX1. If EX1 do not complete the execution, next state will be EX2 and so on.

Multiple- Cycle Hardwired Control

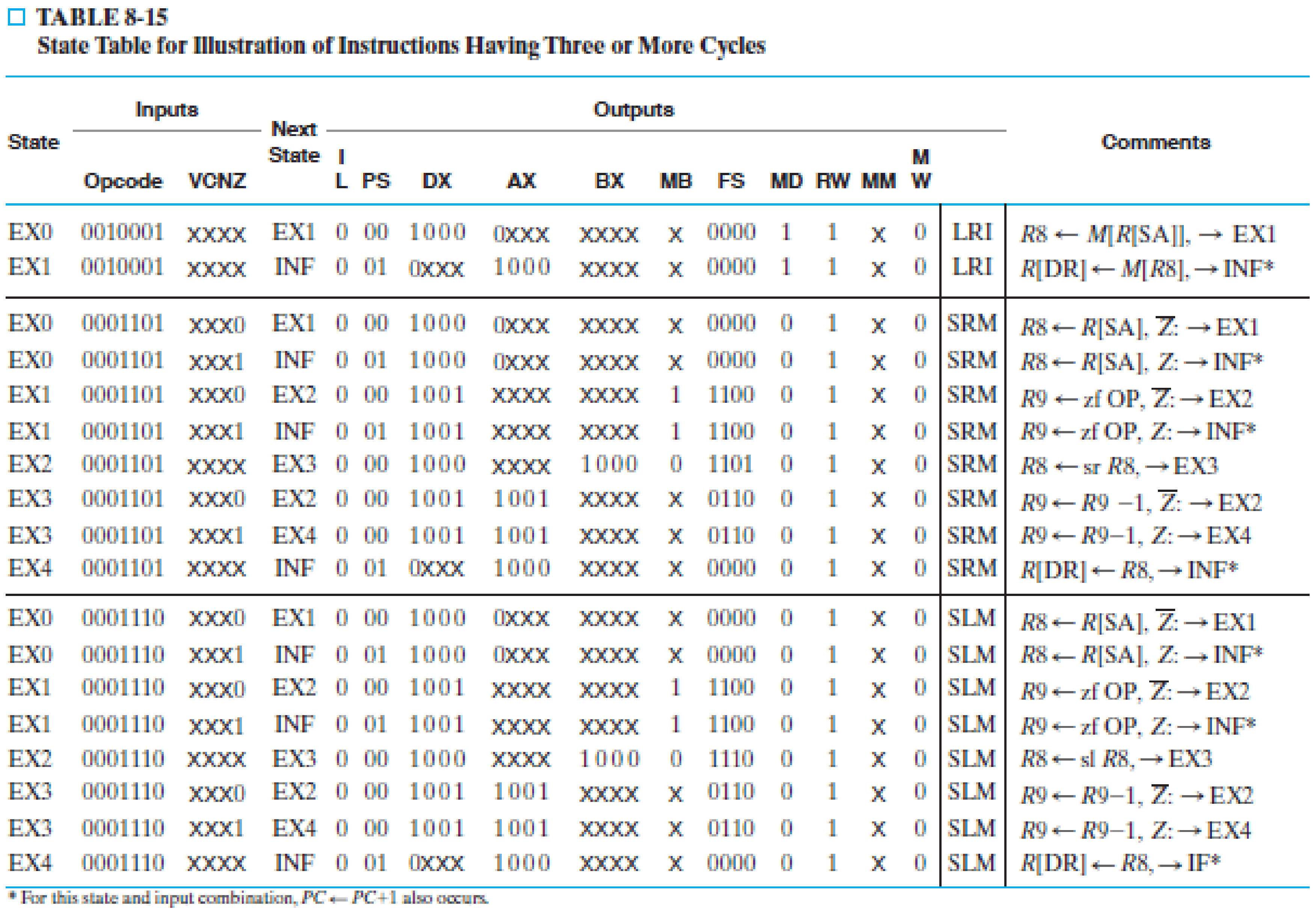


TABLE 8-15
State Table for Illustration of Instructions Having Three or More Cycles

State	Inputs		Next State	Outputs												Comments
	Opcode	VCNZ		IL	PS	DX	AX	BX	MB	FS	MD	RW	MM	MW		
EX0	0010001	xxxx	EX1	0	00	1000	0xxx	xxxx	x	0000	1	1	x	0	LRI	$R8 \leftarrow M[R[SA]], \rightarrow$ EX1
EX1	0010001	xxxx	INF	0	01	0xxx	1000	xxxx	x	0000	1	1	x	0	LRI	$R[DR] \leftarrow M[R8], \rightarrow$ INF*
EX0	0001101	xxx0	EX1	0	00	1000	0xxx	xxxx	x	0000	0	1	x	0	SRM	$R8 \leftarrow R[SA], \overline{Z}: \rightarrow$ EX1
EX0	0001101	xxx1	INF	0	01	1000	0xxx	xxxx	x	0000	0	1	x	0	SRM	$R8 \leftarrow R[SA], Z: \rightarrow$ INF*
EX1	0001101	xxx0	EX2	0	00	1001	xxxx	xxxx	1	1100	0	1	x	0	SRM	$R9 \leftarrow$ zf OP, $\overline{Z}: \rightarrow$ EX2
EX1	0001101	xxx1	INF	0	01	1001	xxxx	xxxx	1	1100	0	1	x	0	SRM	$R9 \leftarrow$ zf OP, $Z: \rightarrow$ INF*
EX2	0001101	xxxx	EX3	0	00	1000	xxxx	1000	0	1101	0	1	x	0	SRM	$R8 \leftarrow$ sr $R8, \rightarrow$ EX3
EX3	0001101	xxx0	EX2	0	00	1001	1001	xxxx	x	0110	0	1	x	0	SRM	$R9 \leftarrow R9 - 1, \overline{Z}: \rightarrow$ EX2
EX3	0001101	xxx1	EX4	0	00	1001	1001	xxxx	x	0110	0	1	x	0	SRM	$R9 \leftarrow R9 - 1, Z: \rightarrow$ EX4
EX4	0001101	xxxx	INF	0	01	0xxx	1000	xxxx	x	0000	0	1	x	0	SRM	$R[DR] \leftarrow R8, \rightarrow$ INF*
EX0	0001110	xxx0	EX1	0	00	1000	0xxx	xxxx	x	0000	0	1	x	0	SLM	$R8 \leftarrow R[SA], \overline{Z}: \rightarrow$ EX1
EX0	0001110	xxx1	INF	0	01	1000	0xxx	xxxx	x	0000	0	1	x	0	SLM	$R8 \leftarrow R[SA], Z: \rightarrow$ INF*
EX1	0001110	xxx0	EX2	0	00	1001	xxxx	xxxx	1	1100	0	1	x	0	SLM	$R9 \leftarrow$ zf OP, $\overline{Z}: \rightarrow$ EX2
EX1	0001110	xxx1	INF	0	01	1001	xxxx	xxxx	1	1100	0	1	x	0	SLM	$R9 \leftarrow$ zf OP, $Z: \rightarrow$ INF*
EX2	0001110	xxxx	EX3	0	00	1000	xxxx	1000	0	1110	0	1	x	0	SLM	$R8 \leftarrow$ sl $R8, \rightarrow$ EX3
EX3	0001110	xxx0	EX2	0	00	1001	1001	xxxx	x	0110	0	1	x	0	SLM	$R9 \leftarrow R9 - 1, \overline{Z}: \rightarrow$ EX2
EX3	0001110	xxx1	EX4	0	00	1001	1001	xxxx	x	0110	0	1	x	0	SLM	$R9 \leftarrow R9 - 1, Z: \rightarrow$ EX4
EX4	0001110	xxxx	INF	0	01	0xxx	1000	xxxx	x	0000	0	1	x	0	SLM	$R[DR] \leftarrow R8, \rightarrow$ IF*

* For this state and input combination, $PC \leftarrow PC+1$ also occurs.