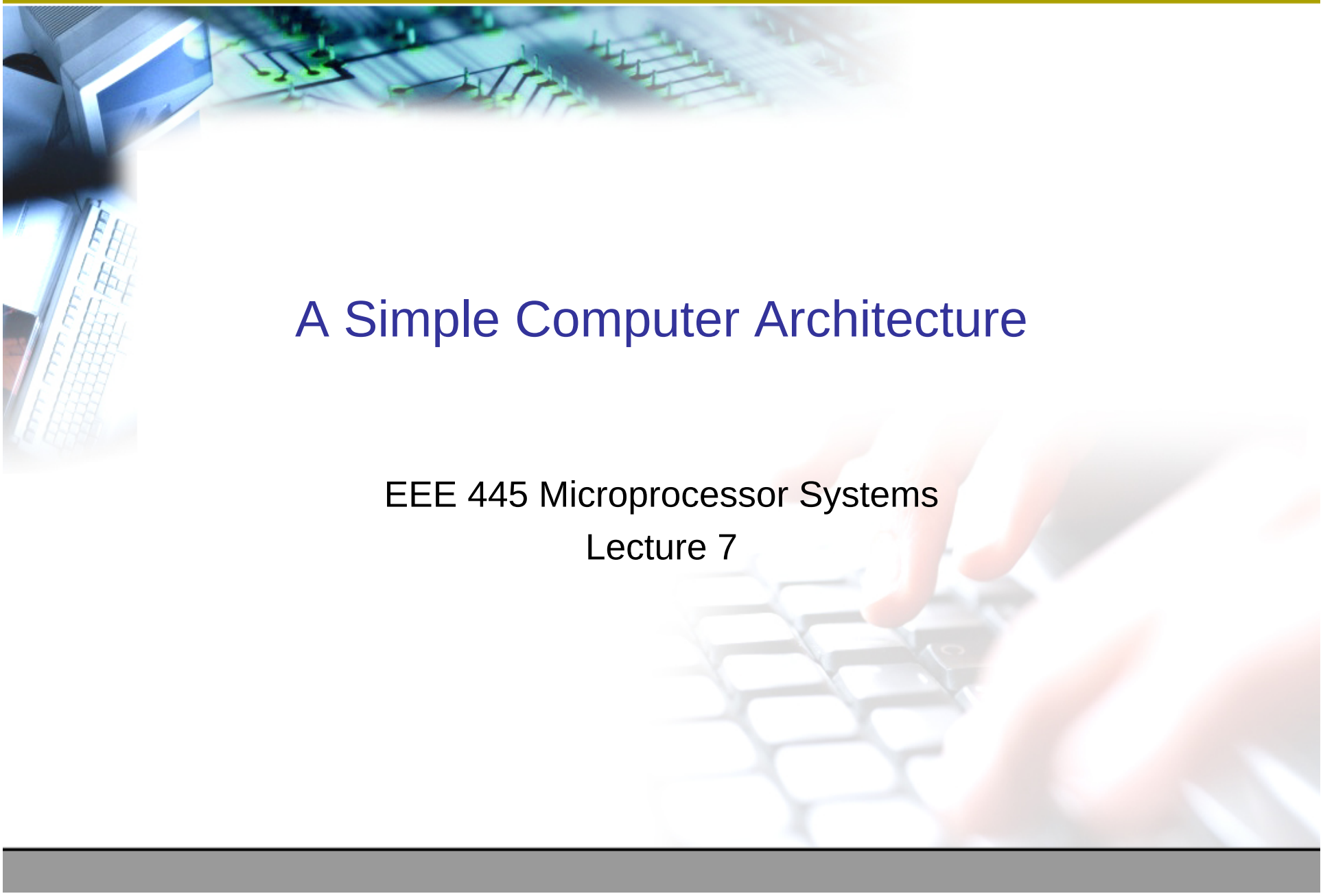


A Simple Computer Architecture

EEE 445 Microprocessor Systems

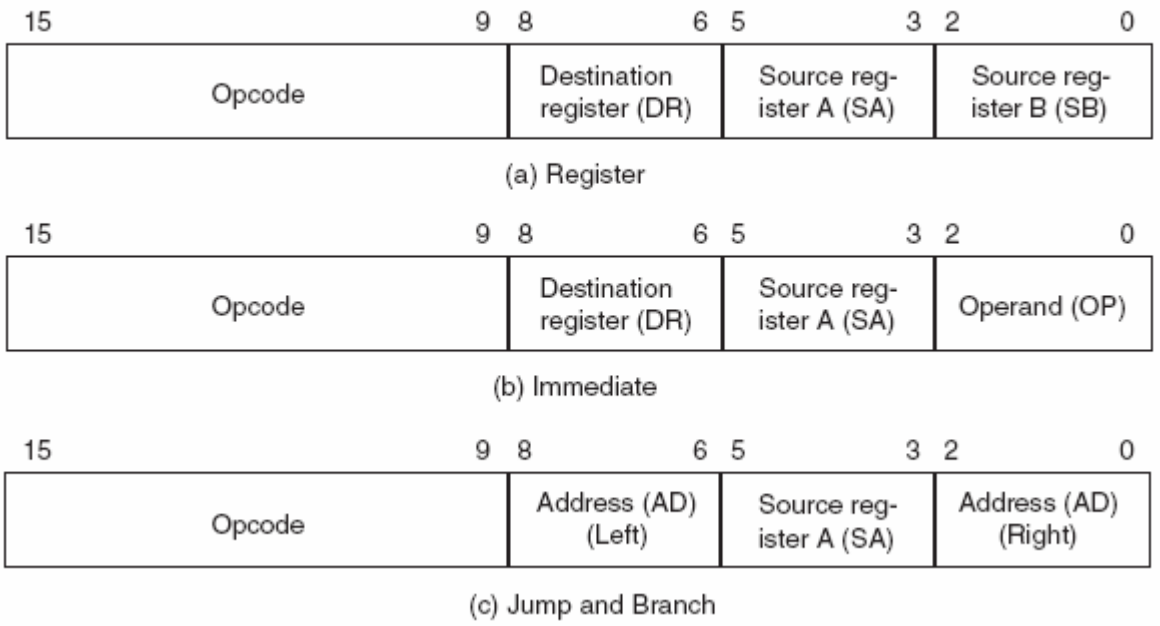
Lecture 7

Instructions

- **An instruction is a collection of bits that instructs the computer to perform a specific operation.**
- **Operation code (opcode) is the group of bits in the instruction that specifies the operation.**
- **Instruction set is the collection instructions.**
- **Instruction set architecture describes the instruction set.**
- **The number of bits of opcode determines the number of distinct operations that can be performed.**
- **Instruction must also specify the operands on which the operations are performed. This can be done explicitly of implicitly.**

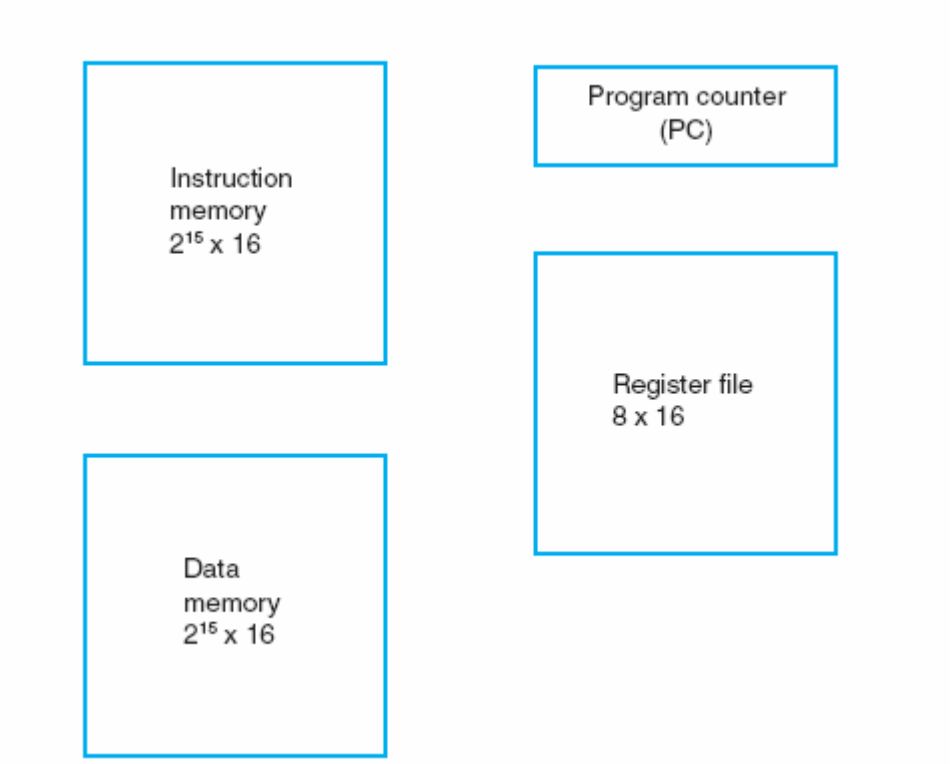
Instruction Formats

- **Each instruction contains fields assigned to a specific item, such as opcode, a constant value or a register file address.**



Storage Resource Diagram for a Simple Computer

- **Visible from programmer**



Instruction Specifications

Instruction Specifications for the Simple Computer

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] ← R[SA] ⊕ R[SB]	N, Z
NOT	0001011	NOT	RD,RA	R[DR] ← $\overline{\text{R[SA]}}$	N, Z
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]	

Memory Representation of Instructions and Data

Memory Representation of Instructions and Data

Decimal 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.		

Block Diagram for a Single-Cycle Computer

- **Hardwired control**
- **Fetches and executes an instruction in a single clock cycle**
- **Bit 13 → jump or conditional jump**
- **Bits 11-9→ condition**

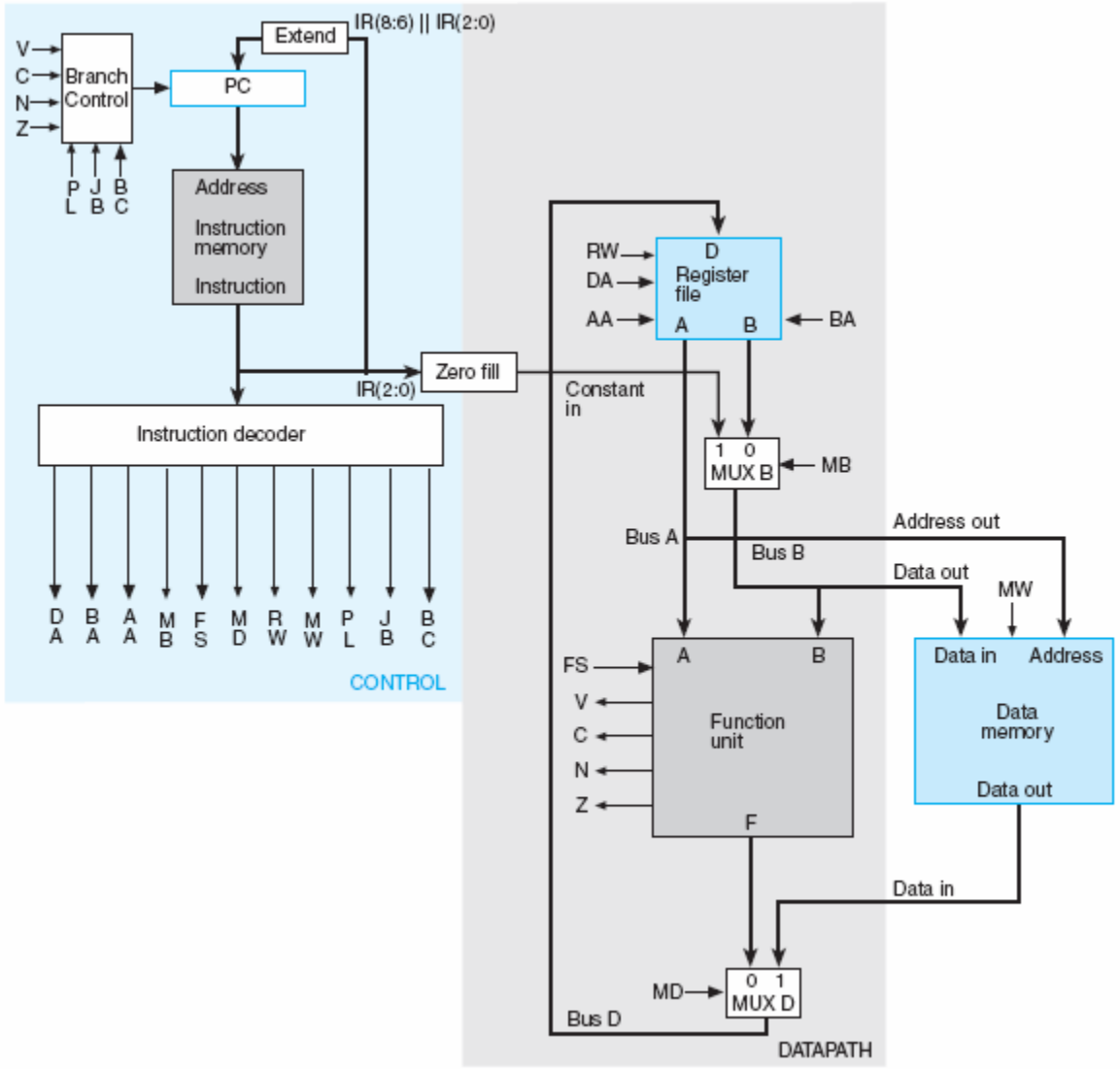
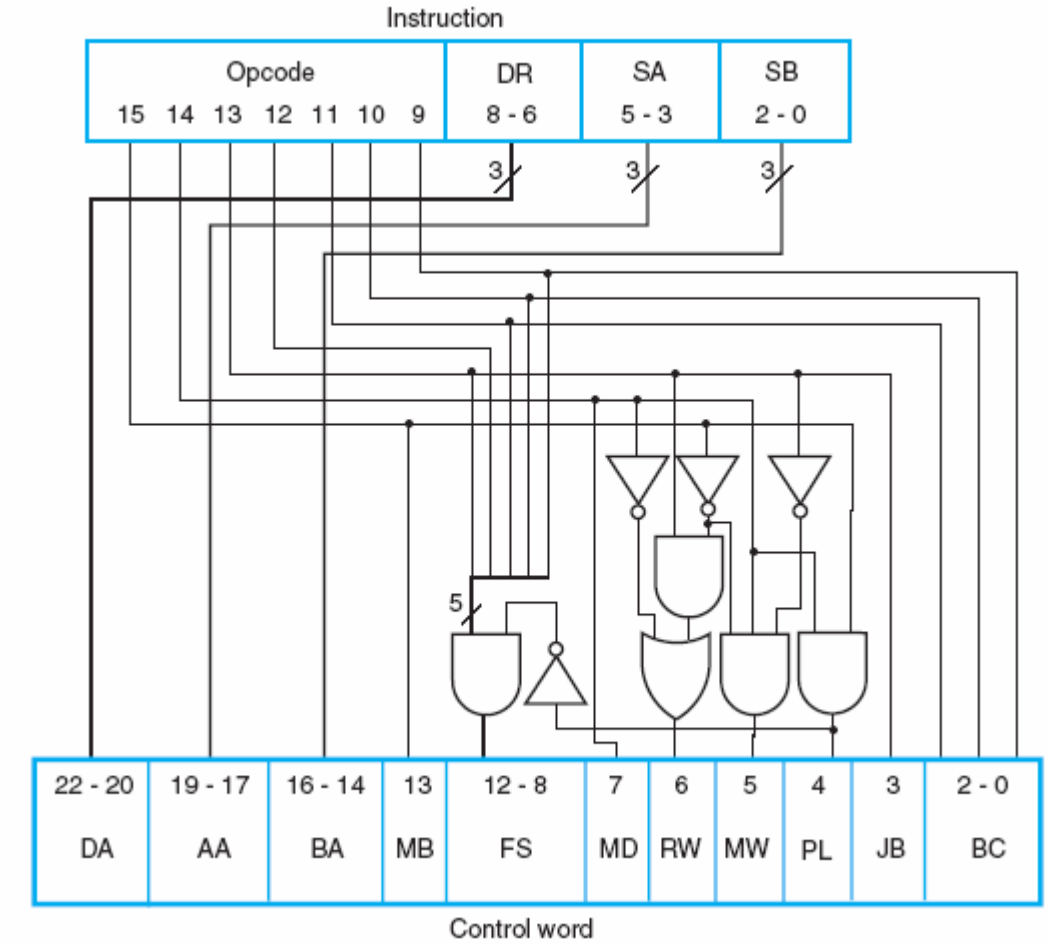


Diagram of Instruction Decoder

- **Instruction decoder determines the control words for the datapath.**



Truth Table for Instruction Decoder Logic

Truth Table for Instruction Decoder Logic

	Instruction Bits			Control Word Bits					
Instruction Function Type	Bit 15	Bit 14	Bit 13	MB	MD	RW	MW	PL	JB
ALU function using registers	0	0	0	0	0	1	0	0	X
Shifter function using registers	0	0	1	0	0	1	0	0	X
Memory write using register data	0	1	0	0	X	0	1	0	X
Memory read using register data	0	1	1	0	1	1	0	0	X
ALU operation using a constant	1	0	0	1	0	1	0	0	X
Shifter function using a constant	1	0	1	1	0	1	0	0	X
Conditional Branch	1	1	0	X	X	0	0	1	0
Unconditional Jump	1	1	1	X	X	0	0	1	1

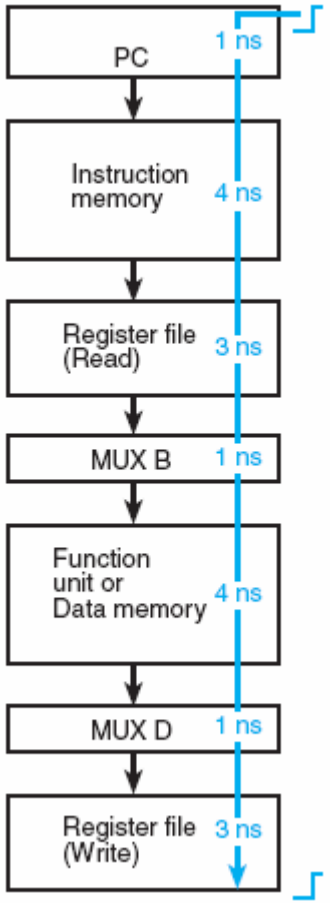
Six Instructions for the Single-Cycle Computer

Six Instructions for the Single-Cycle Computer

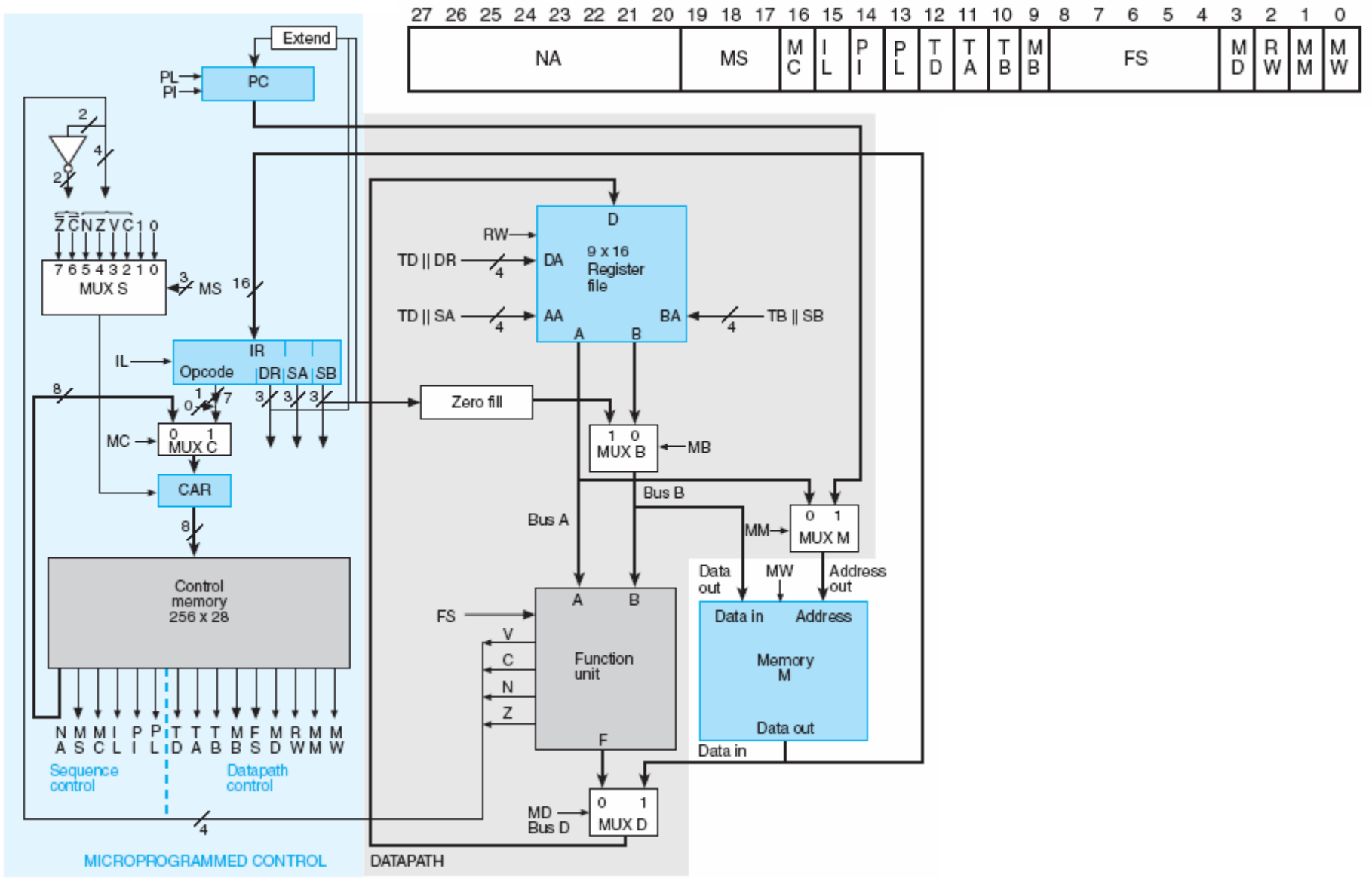
Operation code	Symbolic name	Format	Description	Function	MB	MD	RW	MW	PL	JB
1000010	ADI	Immediate	Add immediate operand	$R[\text{DR}] \leftarrow R[\text{SA}] + \text{zf}\ I(2{:}0)$	1	0	1	0	0	0
0110000	LD	Register	Load memory content into register	$R[\text{DR}] \leftarrow M[R[\text{SA}]]$	0	1	1	0	0	1
0100000	ST	Register	Store register content in memory	$M[R[\text{SA}]] \leftarrow R[\text{SB}]$	0	1	0	1	0	0
0011000	SL	Register	Shift left	$R[\text{DR}] \leftarrow \text{sl}R[\text{SB}]$	0	0	1	0	0	1
0001110	NOT	Register	Complement register	$R[\text{DR}] \leftarrow \overline{R[\text{SA}]}$	0	0	1	0	0	0
1100000	BRZ	Jump/Branch	If $R[\text{SA}] = 0$, branch to PC + se AD	If R[SA] = $0, PC \leftarrow PC + \text{se}AD$, If R[SA] $\neq 0, PC \leftarrow PC + 1$	1	0	0	0	1	0

Problems in Single Cycle Computer

- **Performing complex operations**
- **The usage of the same memory for both data and instructions**
- **Worst Case Delay Path in Single-Cycle Computer**



Multiple-Cycle Microprogrammed Computer



Control Word Information for Datapath and Sequence Control

Control Word Information for Datapath

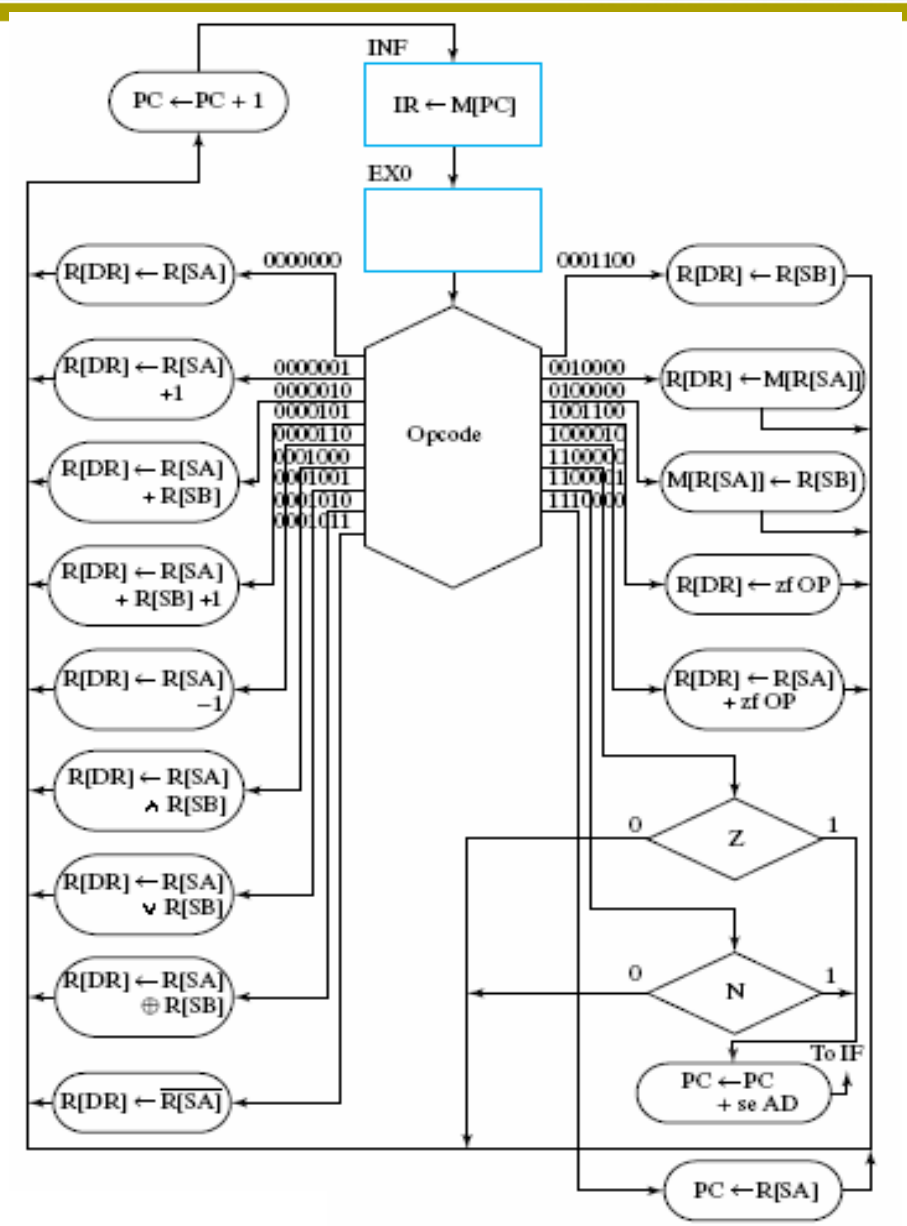
TD	TA	TB	MB		FS		MD	RW	MM	MW	
Select	Select	Select	Select	Code	Function	Code	Select	Function	Select	Function	Code
R[DR]	R[SA]	R[SB]	Register	0	$F = A$	00000	FnUt	No write (NW)	Address	No write (NW)	0
$R8$	$R8$	$R8$	Constant	1	$F = A + 1$	00001	Data In	Write (WR)	PC	Write (WR)	1
					$F = A + B$	00010					
					$F = A + B + 1$	00011					
					$F = A + \overline{B}$	00100					
					$F = A + \overline{B} + 1$	00101					
					$F = A - 1$	00110					
					$F = A$	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					

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 instr.	1
	Branch	10		
	Jump	11		

ASM Chart for Multiple-Cycle Hardwired Computer

- **IF: Instruction fetch**
- **2^7=128 locations in control memory**
- **128 decision boxes**



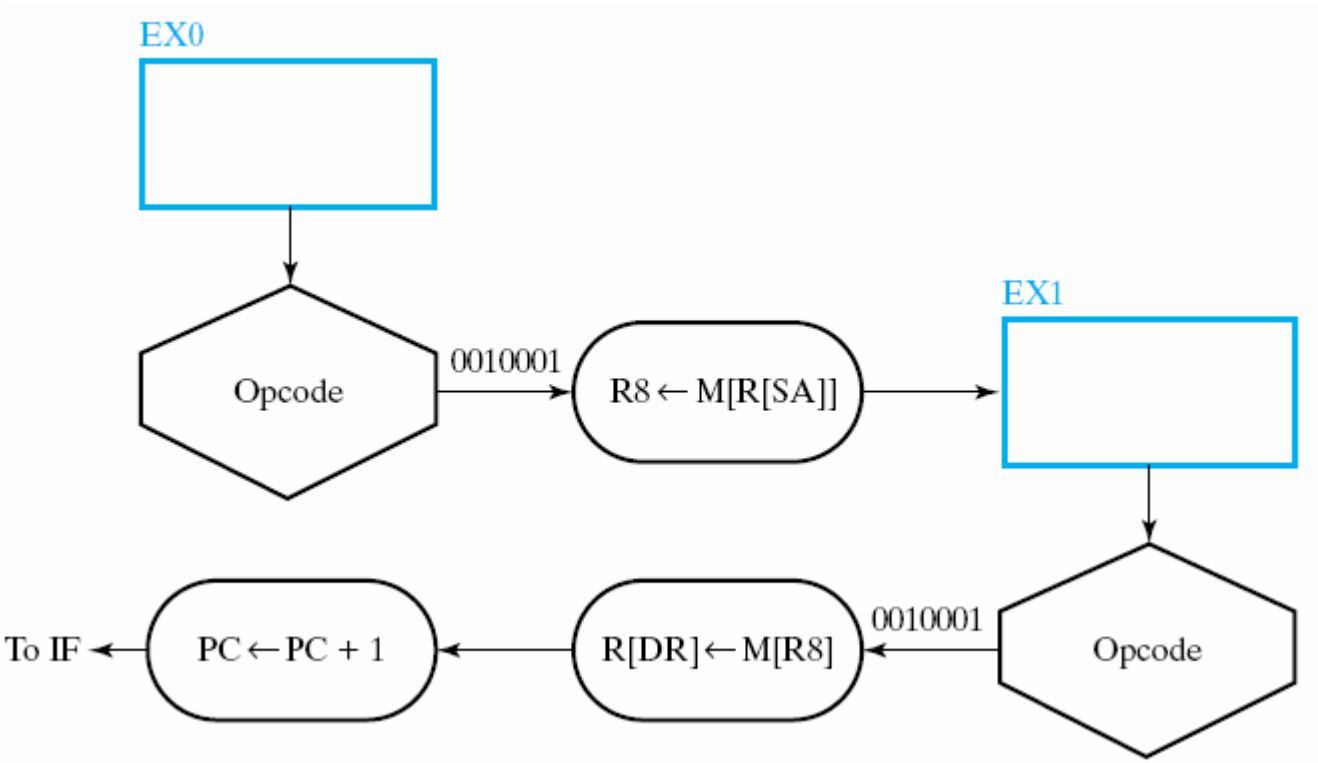
State Table for Two Cycle Instructions

State Table for Two-Cycle Instructions

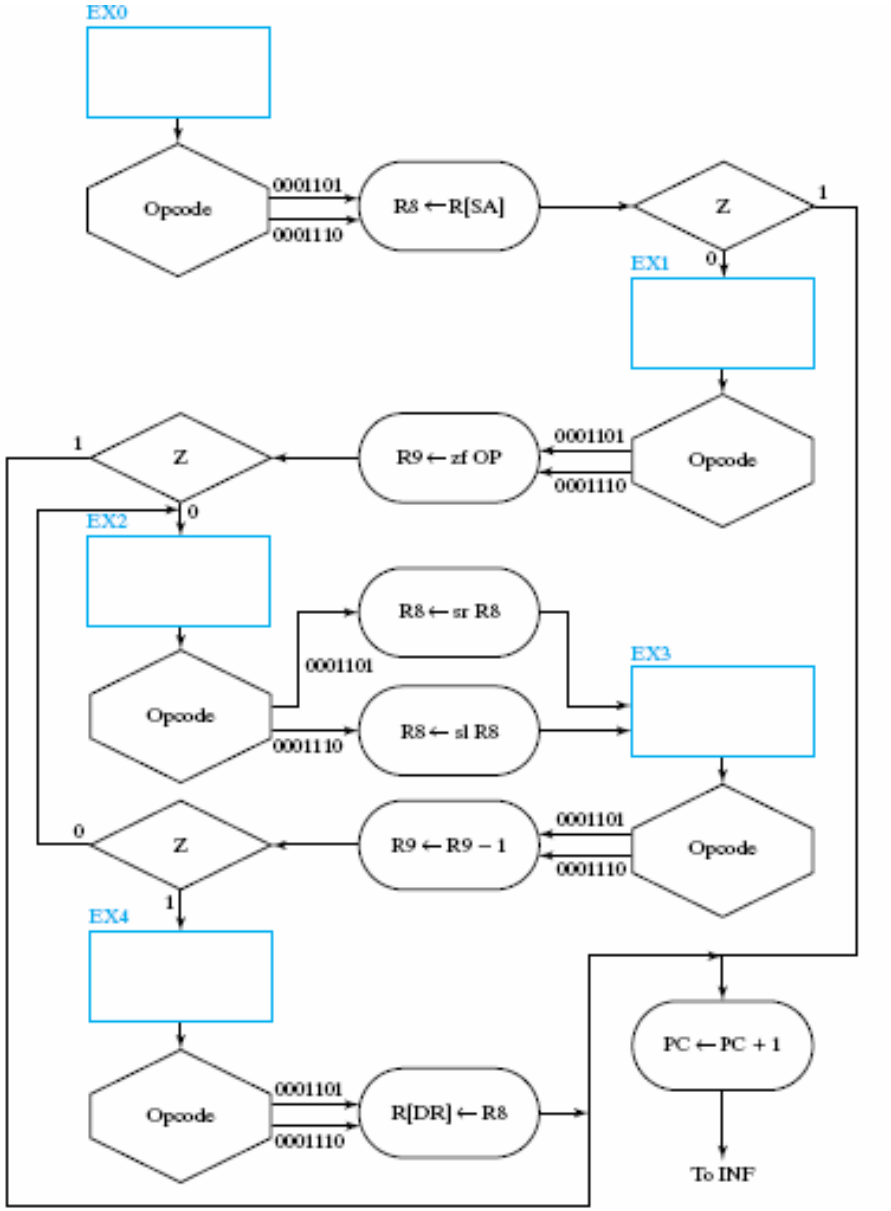
State	Inputs		Next state	Outputs												Comments
	Opcode	VCNZ		IL	PS	DX	AX	BX	MB	FS	MD	RW	MM	MW		
INF	XXXXXXX	XXXX	EX0	1	00	XXXX	XXXX	XXXX	X	XXXX	X	0	1	0		$IR \leftarrow M[PC]$
EX0	0000000	XXXX	INF	0	01	0XXX	0XXX	XXXX	X	0000	0	1	X	0	MOVA	$R[DR] \leftarrow R[SA]$*
EX0	0000001	XXXX	INF	0	01	0XXX	0XXX	XXXX	X	0001	0	1	X	0	INC	$R[DR] \leftarrow R[SA] + 1$*
EX0	0000010	XXXX	INF	0	01	0XXX	0XXX	0XXX	0	0010	0	1	X	0	ADD	$R[DR] \leftarrow R[SA] + R[SB]$*
EX0	0000101	XXXX	INF	0	01	0XXX	0XXX	0XXX	0	0101	0	1	X	0	SUB	$R[DR] \leftarrow R[SA] + \overline{R[SB]} + 1$*
EX0	0000110	XXXX	INF	0	01	0XXX	0XXX	XXXX	X	0110	0	1	X	0	DEC	$R[DR] \leftarrow R[SA] + (-1)$*
EX0	0001000	XXXX	INF	0	01	0XXX	0XXX	0XXX	0	1000	0	1	X	0	AND	$R[DR] \leftarrow R[SA] \wedge R[SB]$*
EX0	0001001	XXXX	INF	0	01	0XXX	0XXX	0XXX	0	1001	0	1	X	0	OR	$R[DR] \leftarrow R[SA] \vee R[SB]$*
EX0	0001010	XXXX	INF	0	01	0XXX	0XXX	0XXX	0	1010	0	1	X	0	XOR	$R[DR] \leftarrow R[SA] \oplus R[SB]$*
EX0	0001011	XXXX	INF	0	01	0XXX	0XXX	XXXX	X	1011	0	1	X	0	NOT	$R[DR] \leftarrow \overline{R[SA]}$ *
EX0	0001100	XXXX	INF	0	01	0XXX	XXXX	0XXX	0	1100	0	1	X	0	MOVB	$R[DR] \leftarrow R[SB]$*
EX0	0010000	XXXX	INF	0	01	0XXX	0XXX	XXXX	X	XXXX	1	1	0	0	LD	$R[DR] \leftarrow M[R[SA]]$*
EX0	0100000	XXXX	INF	0	01	XXXX	0XXX	0XXX	0	XXXX	X	0	0	1	ST	$M[R[SA]] \leftarrow R[SB]$*
EX0	1001100	XXXX	INF	0	01	0XXX	XXXX	XXXX	1	1100	0	1	0	0	LDI	$R[DR] \leftarrow$ zf OP*
EX0	1000010	XXXX	INF	0	01	0XXX	0XXX	XXXX	1	0010	0	1	0	0	ADI	$R[DR] \leftarrow R[SA] +$ 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	$PC \leftarrow R[SA]$

* For this state and input combination, PC ← PC + 1 also occurs.

ASM Chart for Register Indirect Instruction



ASM Chart for Right Shift Multiple Instruction



Instructions having 3 or more cycles

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	*R*8←*M*[*R*[SA],→EX1
EX1	0010001	XXXX	INF	0	01	0XXX	1000	XXXX	X	0000	1	1	X	0	LRI	*R*[DR] ← *M*[*R*8],→INF*
EX0	0001101	XXX0	EX1	0	00	1000	0XXX	XXXX	X	0000	0	1	X	0	SRM	*R*8←*R*[SA],Z:→EX1
EX0	0001101	XXX1	INF	0	01	1000	0XXX	XXXX	X	0000	0	1	X	0	SRM	*R*8←*R*[SA],*Z*:→INF*
EX1	0001101	XXX0	EX2	0	00	1001	XXXX	XXXX	1	1100	0	1	X	0	SRM	*R*9← zf OP,Z:→EX2
EX1	0001101	XXX1	INF	0	01	1001	XXXX	XXXX	1	1100	0	1	X	0	SRM	*R*9← zf OP,*Z*:→INF*
EX2	0001101	XXXX	EX3	0	00	1000	XXXX	1000	0	1101	0	1	X	0	SRM	*R*8 ← sr *R*8,→EX3
EX3	0001101	XXX0	EX2	0	00	1001	1001	XXXX	X	0110	0	1	X	0	SRM	*R*9 ← *R*9 − 1,Z:→EX2
EX3	0001101	XXX1	EX4	0	00	1001	1001	XXXX	X	0110	0	1	X	0	SRM	*R*9 ← *R*9 − 1,*Z*:→EX4
EX4	0001101	XXXX	INF	0	01	0XXX	1000	XXXX	X	0000	0	1	X	0	SRM	*R*[DR] ←*R*8,→ INF*
EX0	0001110	XXX0	EX1	0	00	1000	0XXX	XXXX	X	0000	0	1	X	0	SLM	*R*8←*R*[SA],Z:→EX1
EX0	0001110	XXX1	INF	0	00	1000	0XXX	XXXX	X	0000	0	1	X	0	SLM	*R*8←*R*[SA],*Z*:→INF*
EX1	0001110	XXX0	EX2	0	01	1001	XXXX	XXXX	1	1100	0	1	X	0	SLM	*R*9← zf OP,Z:→EX2
EX1	0001110	XXX1	INF	0	01	1001	XXXX	XXXX	1	1100	0	1	X	0	SLM	*R*9← zf OP,*Z*:→INF*
EX2	0001110	XXXX	EX3	0	00	1000	XXXX	1000	0	1110	0	1	X	0	SLM	*R*8 ← sl *R*8,→EX3
EX3	0001110	XXX0	EX2	0	00	1001	1001	XXXX	X	0110	0	1	X	0	SLM	*R*9 ← *R*9 − 1,Z:→EX2
EX3	0001110	XXX1	EX4	0	00	1001	1001	XXXX	X	0110	0	1	X	0	SLM	*R*9 ← *R*9 − 1,*Z*:→EX4
EX4	0001110	XXXX	INF	0	01	0XXX	1000	XXXX	X	0000	0	1	X	0	SLM	*R*[DR] ←*R*8,→ IF*

*For this state and input combination, PC ←PC + 1 also occurs.