Computer Architecture

Complex Instruction Set Computer

Dr. Masood Mir

Week 15

CISC Architecture

- CISC instruction set architectures are characterized by complex instructions that are, that are **not possible to implement using a single-cycle computer or a single pass through a pipeline**.
- A CISC ISA often employs many **number of addressing modes**.
- Further, the CISC ISA often employs **variable-length instructions**.
- The support for **decision making via conditional branching** is also more sophisticated and versatile.
- A basic architecture for a CISC will be presented with the high-performance of a **RISC for simple instructions** and most of the characteristics of a CISC ISA as just described.

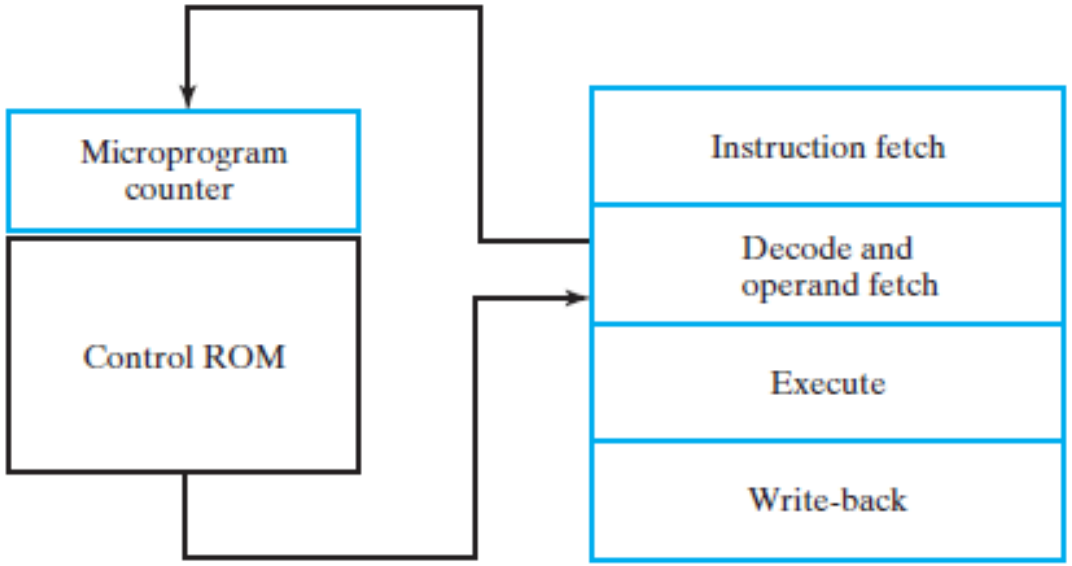

FIGURE 10-18
Combined CISC–RISC Organization

CISC Architecture

- Our approach for CISC architecture is same, we are interested **in approaching a throughput of one instruction per short RISC clock cycle** for simple, frequently used instructions.
- To accomplish this goal, we use a pipelined datapath and a combination of pipelined and **micro-programmed control** as shown in Figure 10-18.
- An instruction is fetched into the IR and enters the Decode and Operand Fetch stage. If it is **a simple instruction** that executes completely in a single pass **through the normal RISC pipeline**, it is decoded and operand fetch occurs as usual.
- On the other hand, if the instruction requires multiple microoperations or multiple memory accesses in sequence, the **decode stage produces an address for the microcode ROM** and replaces the usual instruction decoder outputs with control values from the microcode ROM.

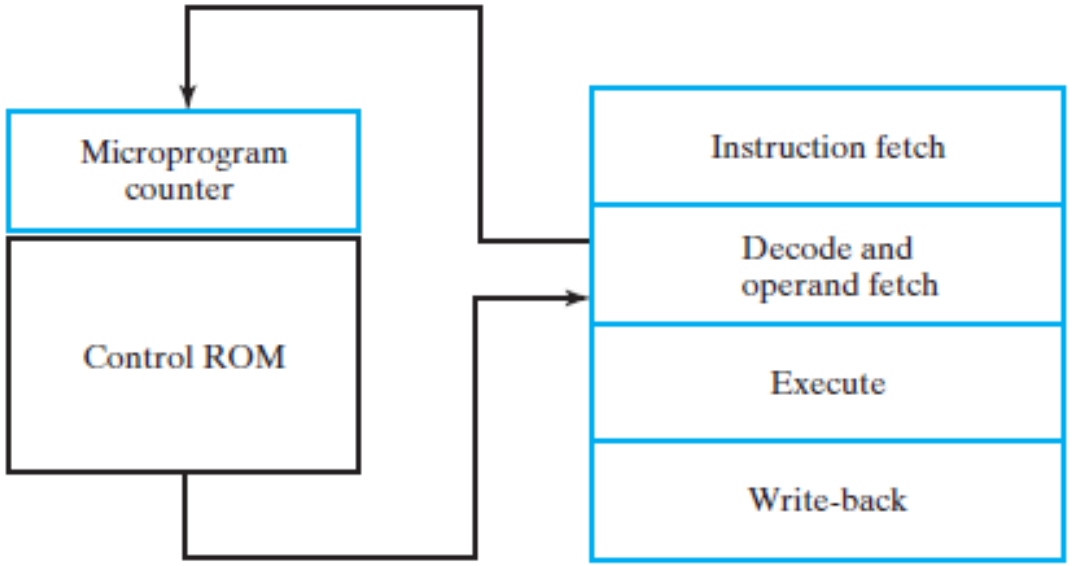

FIGURE 10-18
Combined CISC–RISC Organization

CISC Architecture

- The **microprogram counter** keep providing the addresses for **Control ROM** through multiple clock cycles. The Control ROM produce the micro-instructions for every cycle which is then executed.
- Execution of microinstructions from the ROM, selected by the microprogram counter, continues until the execution of the instruction is completed.
- **Pipelines and microprograms are compatible** and need not be viewed as mutually exclusive.
- The most frequent use of such a combined architecture allows existing software designed for a CISC to take advantage of a RISC architecture while preserving the existing ISA.
- A sequential control of considerable complexity is needed, so microprogrammed control is chosen.

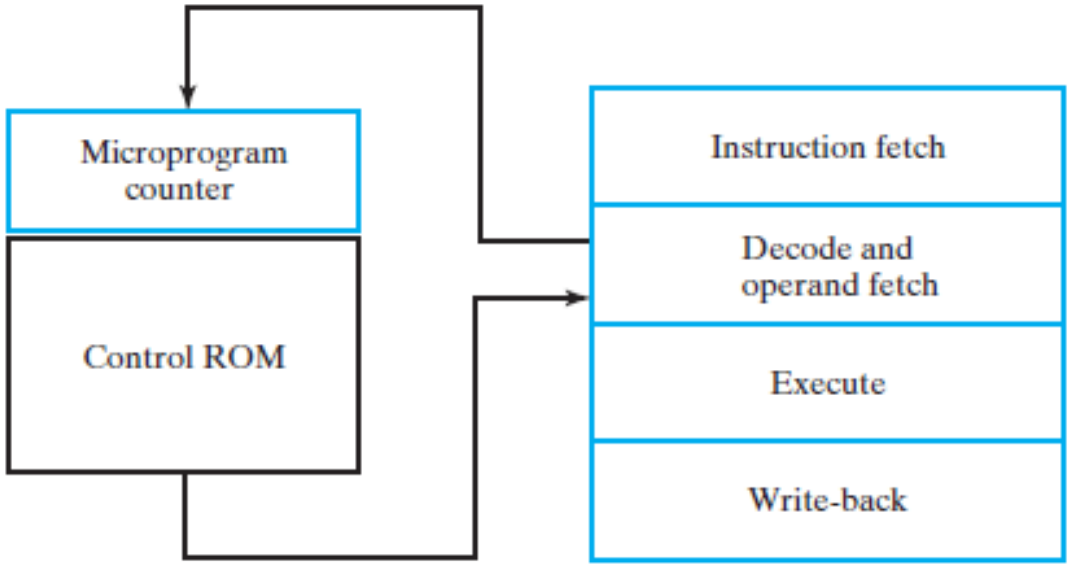

FIGURE 10-18
Combined CISC–RISC Organization

Modifications for CISC Architecture

- The development of the CISC architecture begins with the RISC ISA to obtain some capabilities desirable in the CISC ISA.
- Next, the datapath needs to be modified to support the ISA changes. These include **modification of the Constant Unit**, addition of a **Condition Code register CC**, and deletion of the hardware for supporting the SLT instruction.
- Further, the Register file addressing logic is modified to provide addressing for **16 temporary registers for multiple-pass use of the datapath**, with 16 registers remaining in the storage resources.
- The next step is to adapt the **RISC control to work with the microprogrammed control** in implementing the multiple pass instructions.

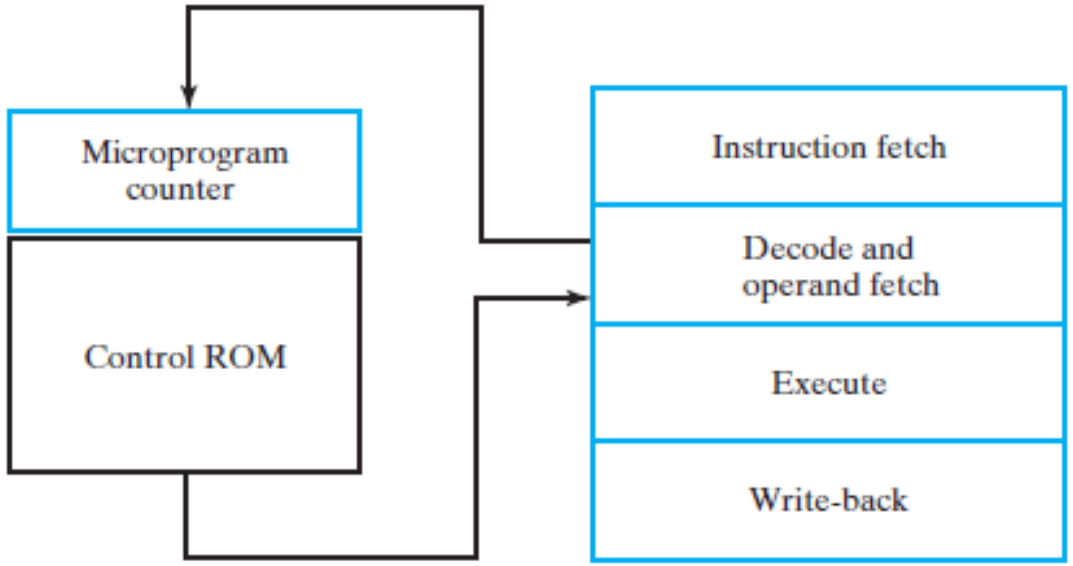

FIGURE 10-18
Combined CISC–RISC Organization

- Finally, the microprogrammed control itself is developed.

ISA Modifications

- The first modification to the RISC ISA is the addition of a **new format for branch instructions**.
- In terms of the instructions provided in the CISC, it is desirable to have the capability to **compare the contents of two source registers and branch**, indicating the relationship between the contents of the two registers.
- To perform such a comparison, a format with two source register fields **SA** and **SB** and a **target offset** are required.
- Addition of the SB field to the branch format reduces the length of the **target offset from 15 bits to 10 bits**.
- The resulting Branch 2 format added for the CISC instructions is shown in Figure 10-19.

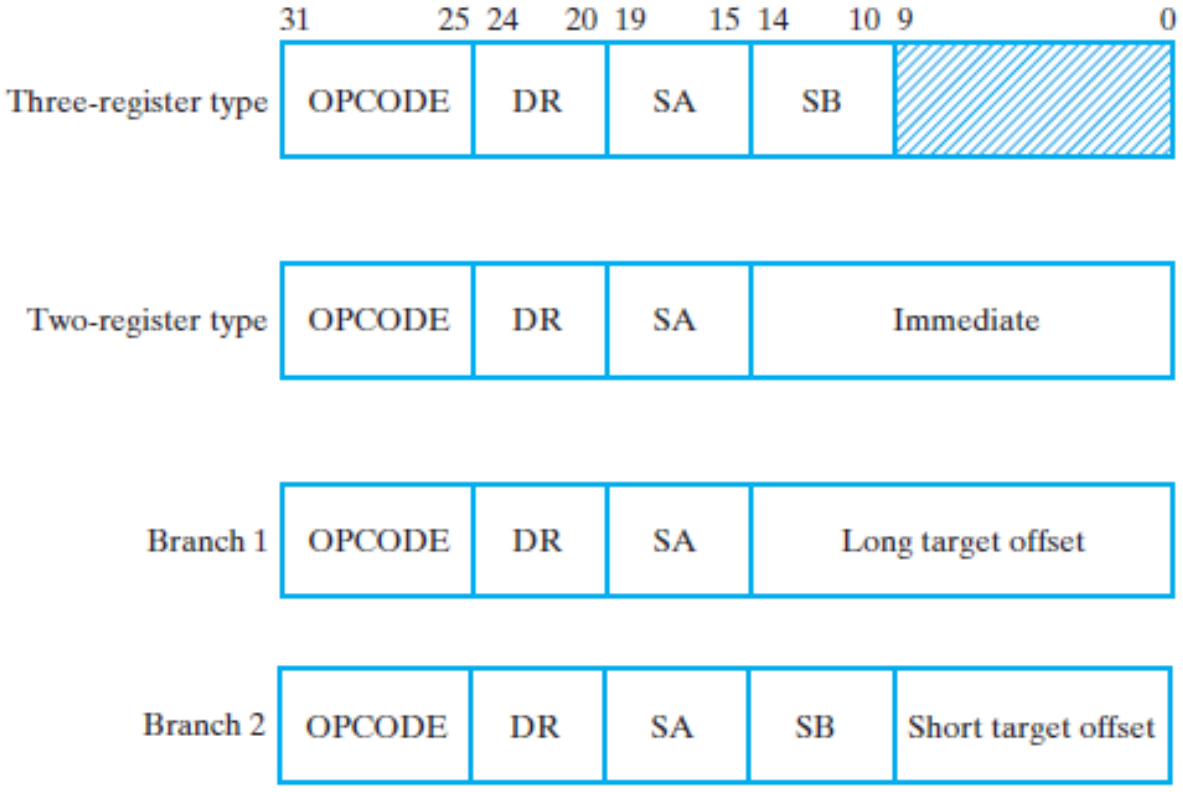

FIGURE 10-19
CISC CPU Instruction Formats

ISA Modifications

- The second modification is to **partition the Register file** to provide addressing for 16 temporary registers for multiple-pass use of the datapath.
- With the partition, only 16 registers remain in the storage resources.
- Rather than modify all of the register address fields in the instruction formats, we will simply **ignore the most significant bit** of these fields for microprogrammed control access of registers.
- For example, only the rightmost four bits of the field DR will be used. DR4 will be ignored.

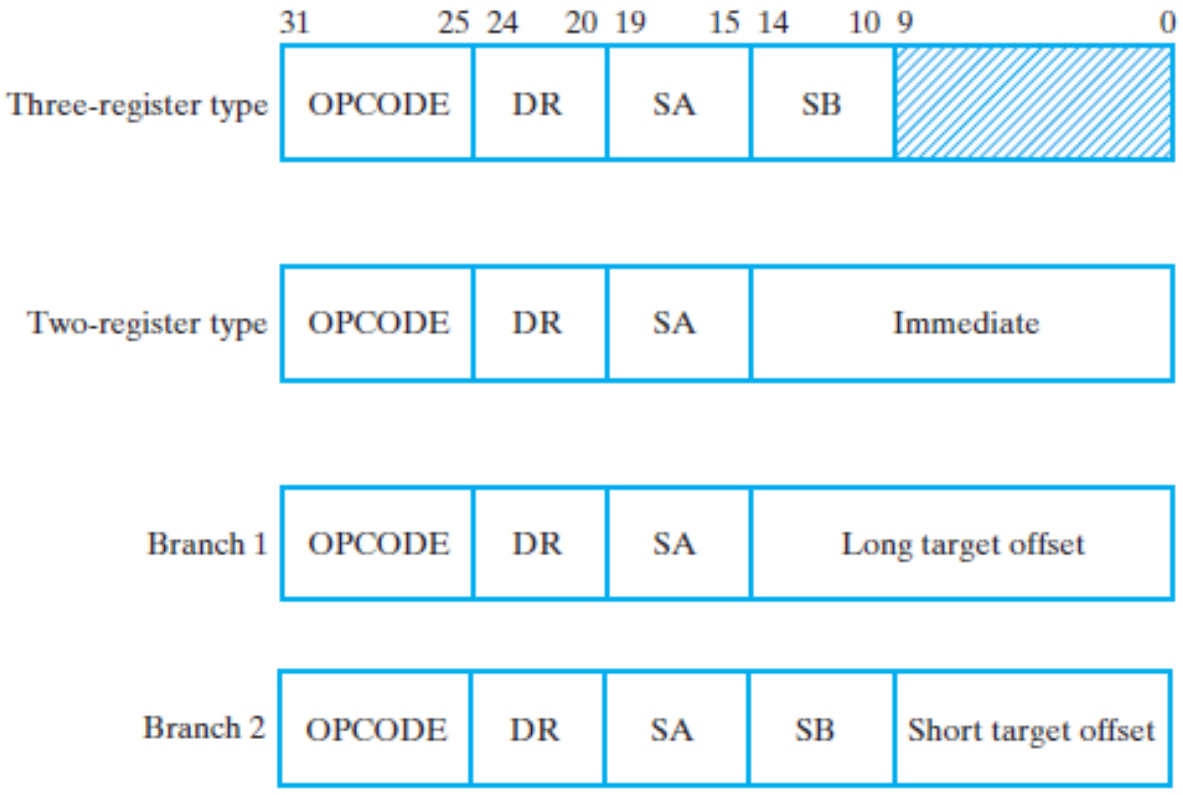

FIGURE 10-19
CISC CPU Instruction Formats

ISA Modifications

- The third modification to the RISC ISA is the **addition of condition codes** (also called flags).
- The condition codes provided are designed specifically to be used in combination with branch on zero and branch on nonzero in implementing instructions that will provide a wide spectrum of decisions, such as **greater than, less than, less than or equal to**, and so on for both **signed and unsigned integers**.
- The **codes are zero (Z), negative (N), carry (C), overflow (V), and less than (L)**. The first four are stored versions of the status outputs of the Function Unit. The **less than (L)** bit is **exclusive OR of Z and V**.
- The inclusion of the L bit in the condition codes eliminates the need for the SLT instruction.

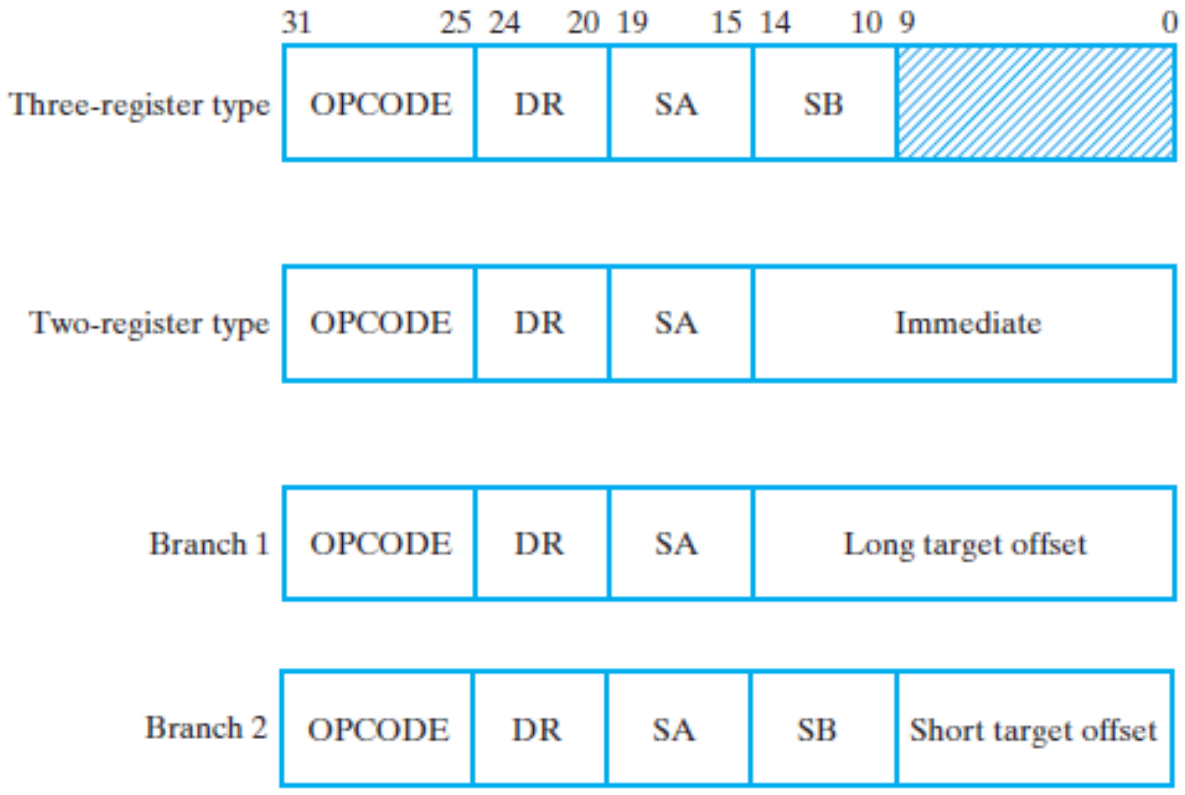

FIGURE 10-19
CISC CPU Instruction Formats

ISA Modifications

- To make the most effective use of these condition codes, it is useful to **control condition codes** whether or not they are modified for a particular micro-operation execution from the instructions.
- Table 10-1 (for RISC instruction codes) shows that bit 4 (third from the left) of the opcode is 0 for the operations MOVA down through instruction LSL.
- This bit can be used for these instructions to control whether the condition codes are affected by the instruction.
- If the **bit4 is 1**, then the **condition-code values are affected** by the execution of the instruction.
- If **bit4 is 0**, then the **condition codes will not be affected**.
- This adds an additional **17 new operation codes** with a 1 in opcode position 4 and 17 new mnemonic codes.

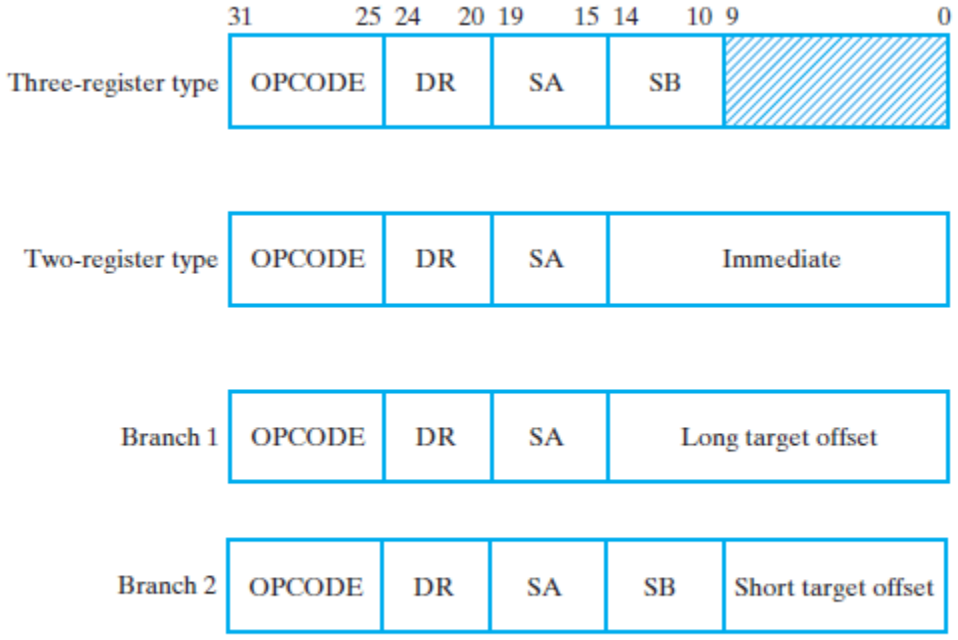

FIGURE 10-19
CISC CPU Instruction Formats

- These opcodes must not overlap the existing operation codes, and the mnemonics are formed by appending C to the current mnemonics in Table 10-1.

Datapath Modifications

- Several changes to the datapath are required to support the ISA modifications.
- First, modifications are made to the Constant unit to handle the change in the length of target offset.
- Logic added to the Constant unit extracts a constant, **IMS = IR9:0, from constant IM**. Sign extension is applied to IMS to obtain a 32-bit word.
- For use in comparisons with condition-code values, an **8-bit constant CA** is provided from the **micro-instruction register, MIR**, in the microprogrammed control.
- This constant is zero filled to form a 32-bit word.
- The **CS** control field for the Constant unit is expanded to two bits to perform selection from among the four possible constant sources.

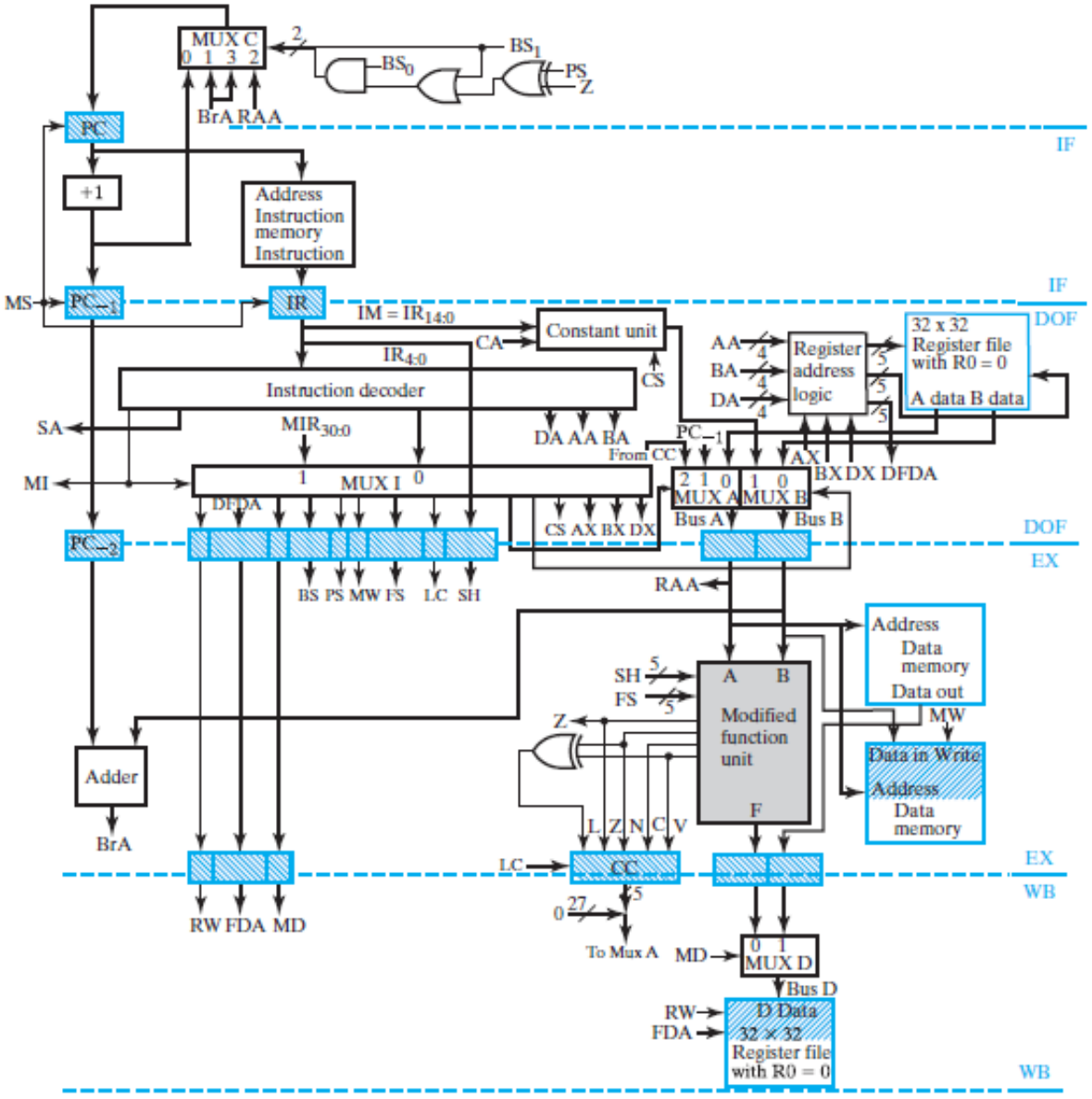

FIGURE 10-20
Pipelined CISC CPU

Datapath Modifications

- Second, the **Register address logic** for multiple-cycle computer is added to the address inputs of the Register file.
- The purpose of this change is to support the ISA modification that provides 16 temporary registers and 16 registers that are a part of the storage resources.
- An additional mode supports the **use of DX as a register-file source address** with BX as the corresponding register-file destination address. This is necessary to capture the contents for R[DR] for use in destination address mode calculations.

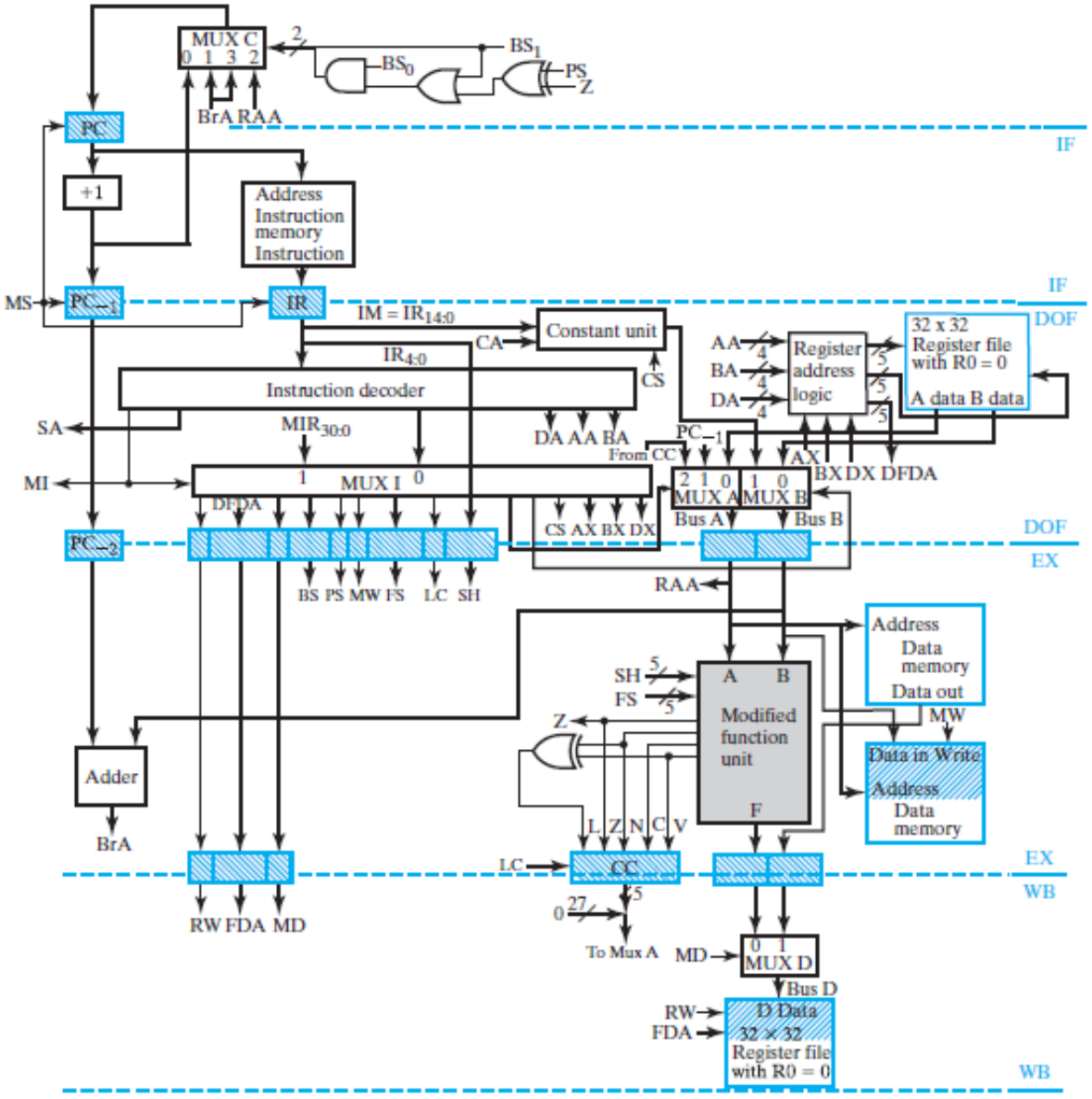

FIGURE 10-20
Pipelined CISC CPU

Datapath Modifications

- Third, a number of changes are made to support the modification adding condition codes.
- In the **DOF stage**, an **additional port is added on MUX A** in order to provide access to CC, the stored condition codes, for storage in temporary registers or comparison to constant values.
- In the **EX stage**, the **condition-code bit L** (less than) is implemented and the condition-code register **CC is added to the pipeline platform**.
- The **new control signal LC** determines whether CC is loaded for the execution of a specific micro-operation using a function unit operation.
- In the **WB stage**, the logic for support of the SLT instruction is replaced by a **zero-filled CC value**, which is passed to the new port on MUX A.

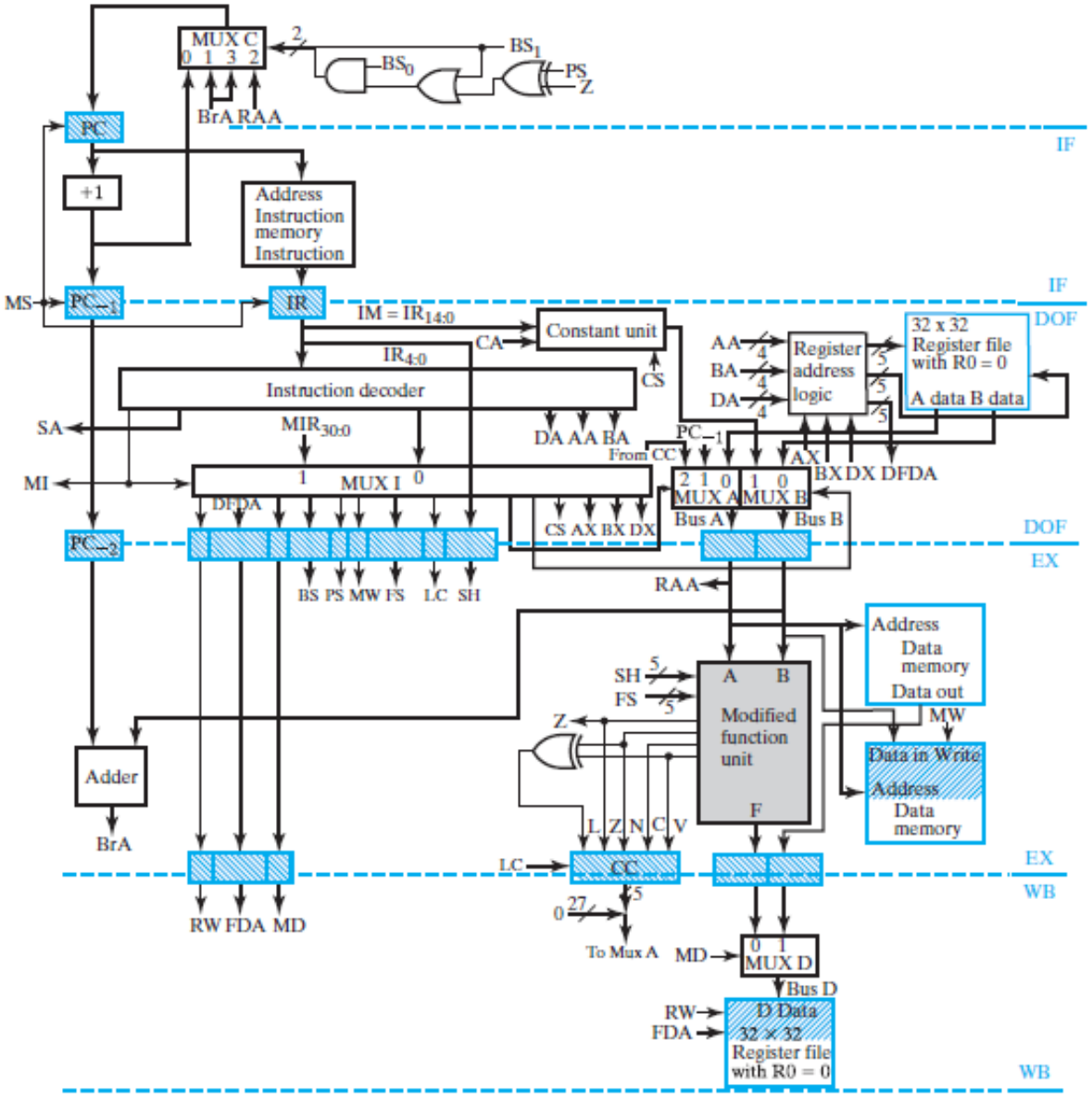

FIGURE 10-20
Pipelined CISC CPU

Control Unit Modifications

- The addition of a microprogrammed control to the control unit to **support instruction implementation using multiple passes** through the pipeline causes significant changes to the existing control, as shown in Figure 10-20.
- The **microprogrammed control is a part of the instruction decoding hardware in the DOF stage**, but it interacts with other parts of the control as well.
- In multiple-pass instruction, the PC points to the instruction in the Instruction memory. The instruction is fetched in the IF stage, and on the next clock edge it is loaded into the IR and the PC is updated. If the **instruction is identified as a multiple-pass instruction from its opcode, microprogrammed Control ROM takes over**.

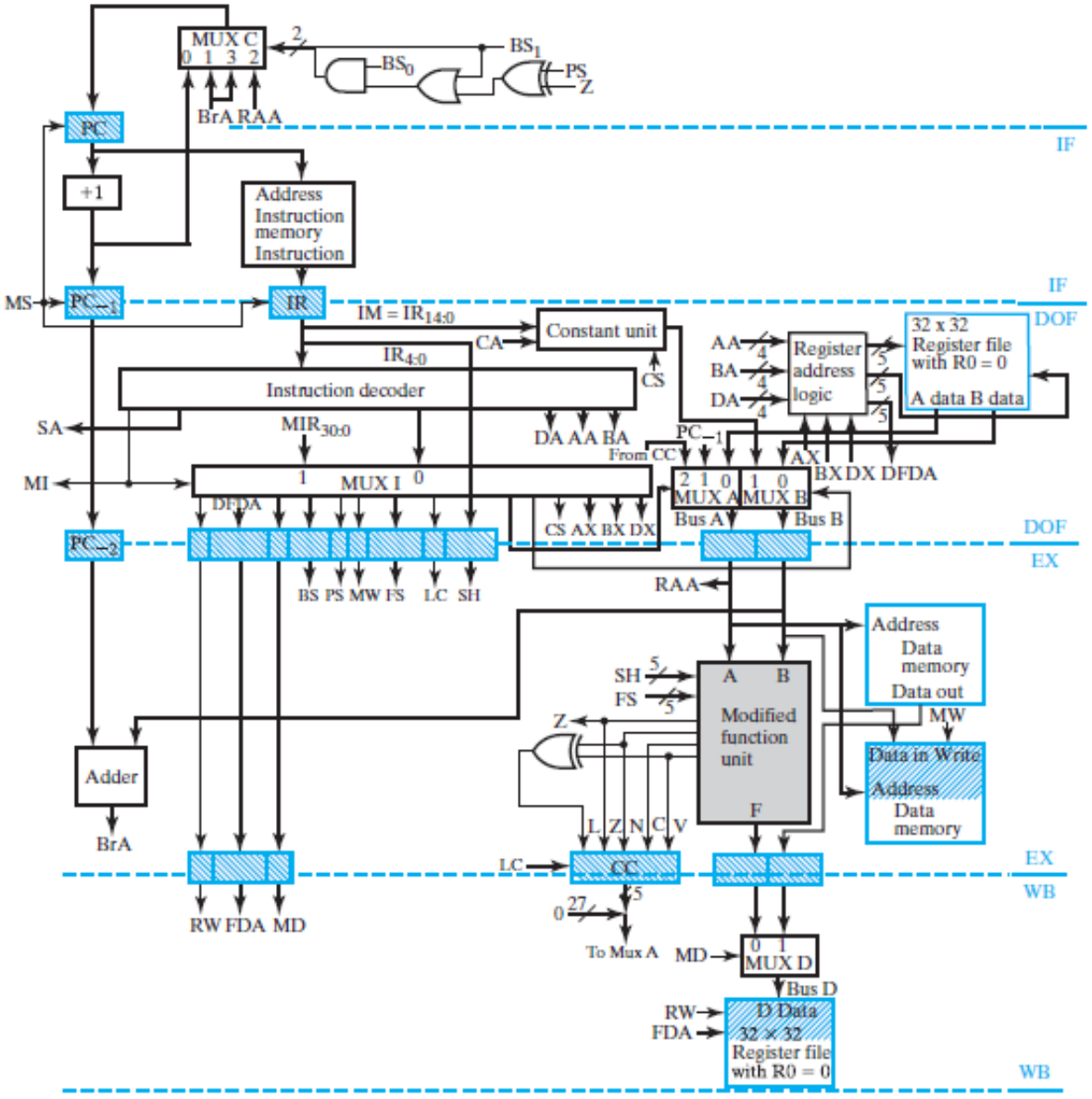

FIGURE 10-20
Pipelined CISC CPU

Control Unit Modifications

- Decoding of the opcode changes **signal MI** to 1 to indicate that this instruction is to use the microprogrammed control. The decoder also produces an **8-bit starting address, SA**, that identifies the beginning of the microprogram in the Microcode ROM.
- Since multiple passes through the pipeline are needed, the loading of subsequent instructions into the IR and further updating of the PC must be prevented. A **signal MS** produced by the microprogrammed control logic becomes 1 and **stalls the PC and the IR**. This prevents the PC from incrementing, but permits PC + 1 to continue down the pipeline into PC_{-1} and PC_{-2} for use in a branch.
- This stall remains until the multiple-pass instruction has been executed or until branch or jump occurs.

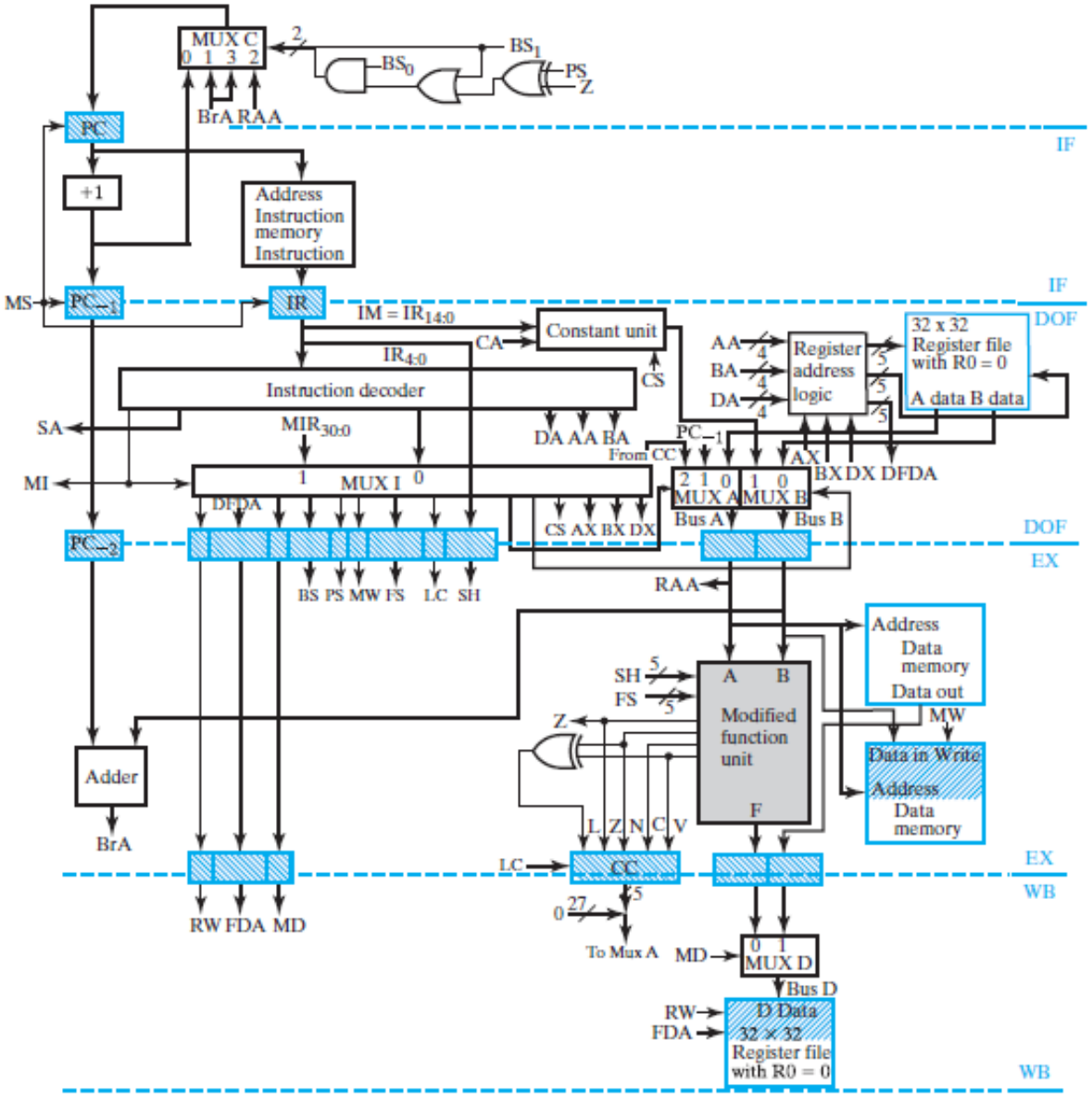

FIGURE 10-20
Pipelined CISC CPU

Control Unit Modifications

- Also, **when MI = 1**, most of the fields of the decoded instruction are replaced with fields of the current microinstruction, which is a decoded NOP (no operation).
- This 31-bit field replacement, performed by **MUX I**, prevents the instruction itself from causing any direct actions.
- Fields **CS** and **MA** have been expanded to two bits each, and field **LC** has been added.
- At this point, the microprogrammed control is now controlling the pipeline and supplies a series of microinstructions (control words) to implement the instruction execution.
- The control word format follows that for the multiple cycle computer and includes fields such as SH, AX, BX, and DX.

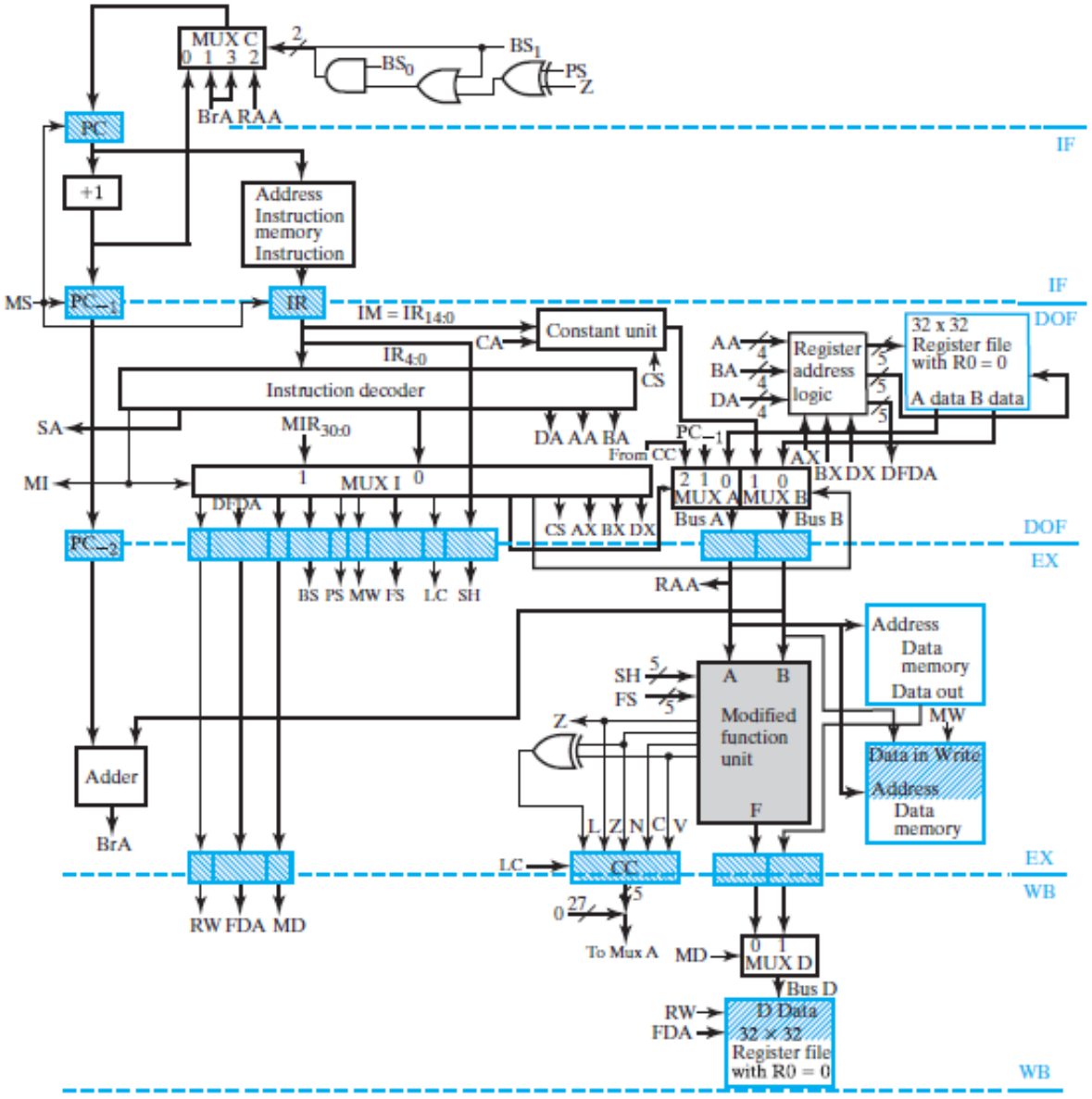

FIGURE 10-20
Pipelined CISC CPU

Control Unit Modifications

- DX is modified to match the register address changes described for the datapath. In addition, the microprogrammed control has to interact with the datapath in order to perform decisions. This interaction includes application of the constant CA, use of the condition codes CC, and use of the zero detect signal Z.
- To support the operations just discussed, the following changes are made to the control unit:

1. Addition of the stall signal MS to the PC, PC_{-1}, IR.
2. Changes in instruction decoder to produce MI, ST.
3. Expansion of the fields CS and MA to two bits.
4. Addition of MUX I, and
5. Addition of control fields AX, BX, and DX, and LC.

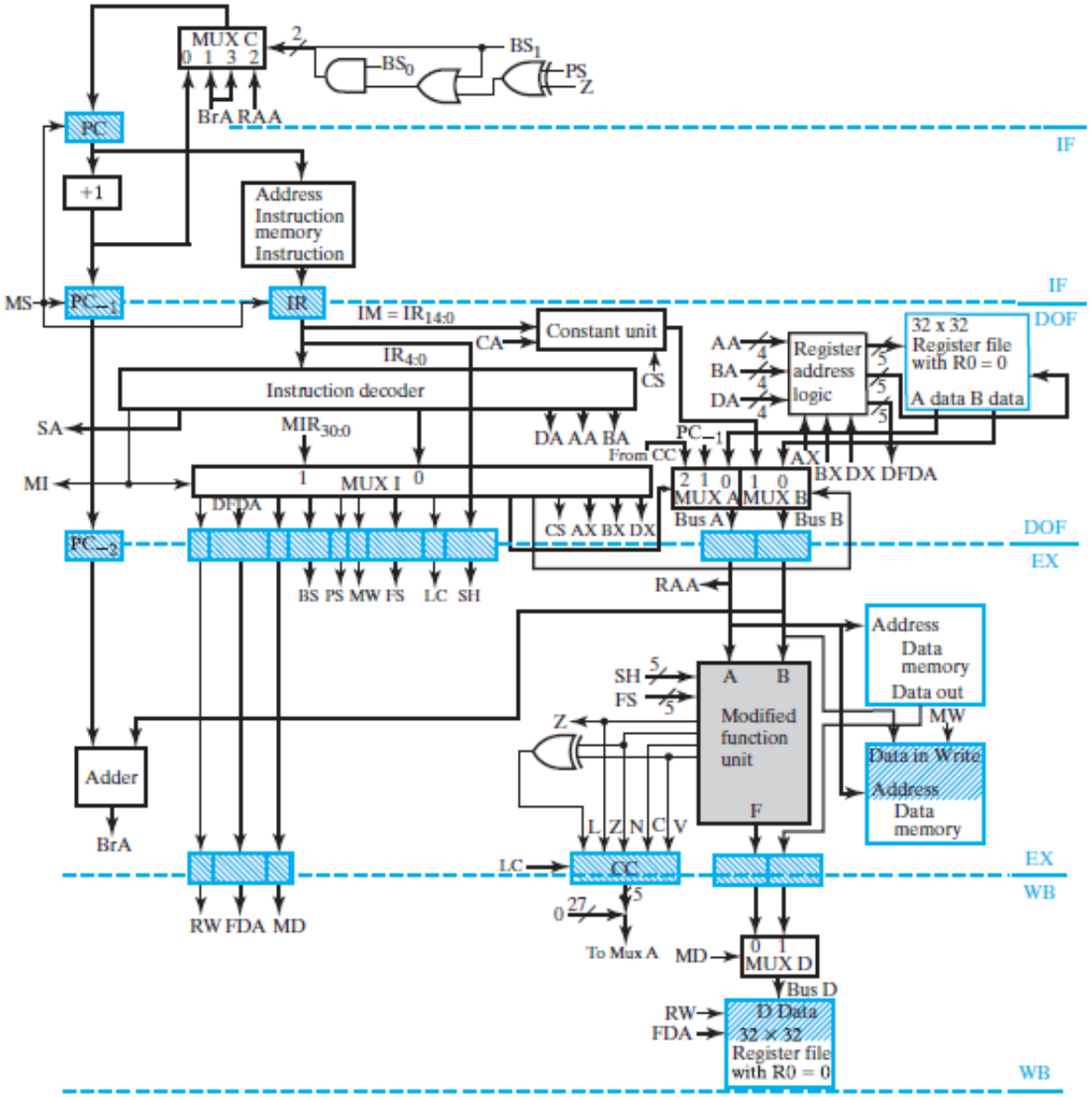

FIGURE 10-20
Pipelined CISC CPU

Control Unit Modifications

- The control is centered about the Microcode ROM, which has an 8-bit address and stores up to 256 41-bit microinstructions.
- The microprogram counter MC stores the address corresponding to the current microinstruction stored in the microinstruction register, MIR.
- The address for the ROM is provided by MUX E, which selects from the incremented MC, the jump address obtained from the microinstruction, CA, the prior value of the jump address, CA-1, and the starting address from the instruction decoder in the control unit, SA.
- Table 10-5 defines the 2-bit select input ME for MUX E and stall bit, MS, in terms of the new control field MZ plus other variables.

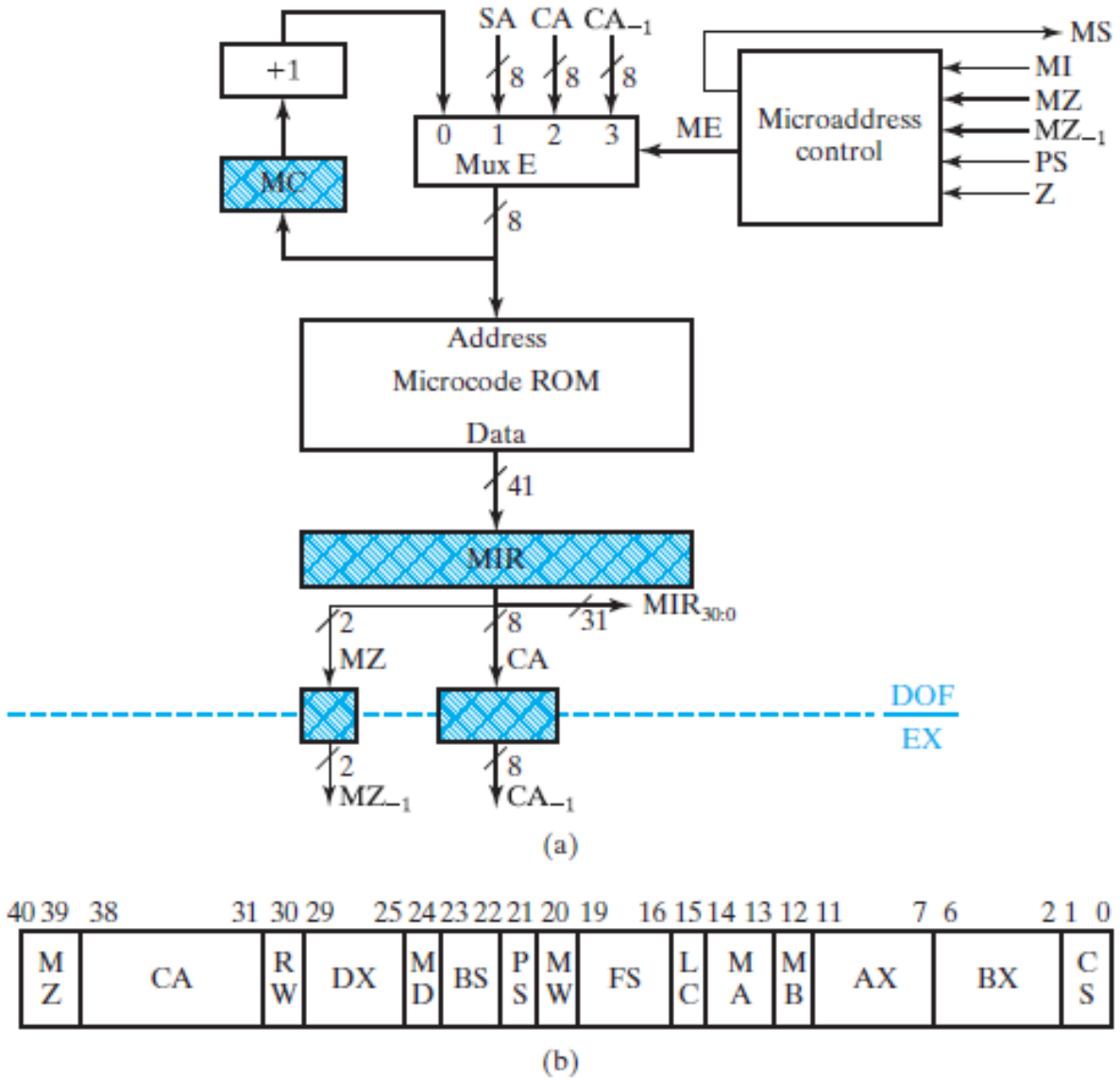

FIGURE 10-21
Pipelined CISC CPU: Microprogrammed Control

Control Unit Modifications

- This function is implemented by the Microaddress Control logic. To set the context for the discussion, in location 0 of the ROM, the IDLE state 0 for the microprogrammed control contains a microinstruction that is a NOP consisting of all zeros. This microinstruction has MZ = 0 and CA = 0.
- From Table 10-5, with MI = 0, the microprogram address is CA = 0, causing the control to remain in this state until MI = 1. With MI = 1, starting address SA is applied to fetch the first microinstruction of the microprogram for the complex instruction being held in IR.
- In the control unit, MI = 1 also switches MUX I from the normal control word coming from the decoder to the 31-bit MIR portion that is a NOP instruction.

TABLE 10-5
Address Control

Inputs					Outputs			
MZ_{-1}	MZ	MI	PS	Z	ME_1	ME_0	MS	Register Transfer Due to ME
11	01	X	0	0	0	0	1	$\overline{PS} \cdot \overline{Z}$: $MC \leftarrow MC + 1$
11	01	X	0	1	0	1	1	$\overline{PS} \cdot Z$: $MC \leftarrow CA_{-1}$
11	01	X	1	0	0	1	1	$PS \cdot \overline{Z}$: $MC \leftarrow CA_{-1}$
11	01	X	1	1	0	0	0	$PS \cdot Z$: $MC \leftarrow MC + 1$
0X	01	X	X	X	0	0	1	$MC \leftarrow MC + 1$
X0	01	X	X	X	0	0	1	$MC \leftarrow MC + 1$
XX	00	0	X	X	1	0	0	$MC \leftarrow CA$
XX	00	1	X	X	0	1	1	$MC \leftarrow ST$
XX	10	X	0	X	1	0	0	$\overline{PS}$: $MC \leftarrow CA$
XX	10	X	1	X	1	0	1	PS: $MC \leftarrow CA$
XX	11	X	X	X	0	0	1	$MC \leftarrow MC + 1$

Control Unit Modifications

- In addition, the output MS from the Microaddress control becomes 1, stalling the PC, PC_{-1}, and the IR in the main control.
- At the next clock edge, the microinstruction fetched from the starting address SA enters the MIR, and the pipeline is now controlled by the microprogram.
- In Figure 10-21, two pipeline registers are required as a part of the microprogrammed control. The stored pipeline values, MZ_{-1} and CA_{-1}, are required for the execution of a conditional microbranch, since the value of Z to be tested occurs during the execution cycle for the microbranch instruction, one clock cycle after it enters the MIR.
- During the execution of the microprogram, the microaddress is controlled by MZ, MZ_{-1}, MI, PS, and Z.

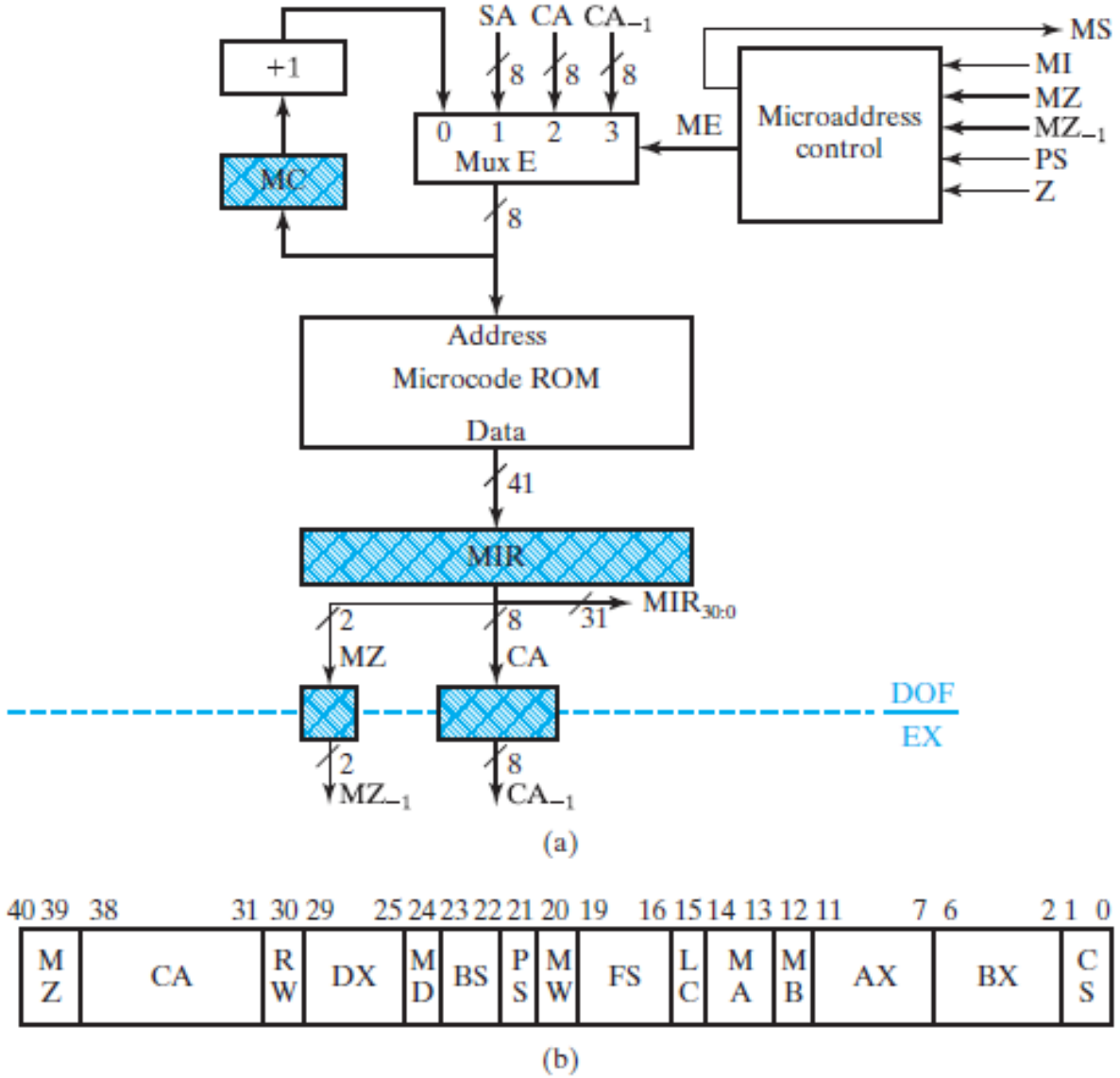

FIGURE 10-21
Pipelined CISC CPU: Microprogrammed Control

Control Unit Modifications

- For MZ_{-1} = 11, MZ = 01 since the microinstruction following a conditional microbranch must be a NOP. Under these conditions, the ME values are controlled by PS and Z with MS = 1.
- For PS and Z having opposite values, a conditional branch to the microaddress value from CA_{-1} occurs.
- Otherwise, for MZ_{-1} = 11 and MZ = 01, the next microaddress becomes the incremented value of MC.
- For MZ−1 ≠ 11, MZ, MI, and PS control the microaddress.
- For MZ = 00, the values of ME and MS are controlled by MI.
- For MI = 0, the next microaddress is CA and MS = 0, corresponding to the idle state for the micro-programmed control.

TABLE 10-5
Address Control

Inputs					Outputs			
MZ_{-1}	MZ	MI	PS	Z	ME_1	ME_0	MS	Register Transfer Due to ME
11	01	X	0	0	0	0	1	$\overline{PS} \cdot \overline{Z}$: $MC \leftarrow MC + 1$
11	01	X	0	1	0	1	1	$\overline{PS} \cdot Z$: $MC \leftarrow CA_{-1}$
11	01	X	1	0	0	1	1	$PS \cdot \overline{Z}$: $MC \leftarrow CA_{-1}$
11	01	X	1	1	0	0	0	$PS \cdot Z$: $MC \leftarrow MC + 1$
0X	01	X	X	X	0	0	1	$MC \leftarrow MC + 1$
X0	01	X	X	X	0	0	1	$MC \leftarrow MC + 1$
XX	00	0	X	X	1	0	0	$MC \leftarrow CA$
XX	00	1	X	X	0	1	1	$MC \leftarrow ST$
XX	10	X	0	X	1	0	0	$\overline{PS}$: $MC \leftarrow CA$
XX	10	X	1	X	1	0	1	PS: $MC \leftarrow CA$
XX	11	X	X	X	0	0	1	$MC \leftarrow MC + 1$

Control Unit Modifications

- For MI = 1, the next microaddress is SA and MS = 1, selecting the next microinstruction from the Microcode ROM and stalling the first two pipeline platforms.
- For MZ = 01, the next microaddress is the incremented value of MC, advancing execution to the next microinstruction in sequence.
- For MZ = 10, an unconditional jump is performed in the microcode control and the value of MS is controlled by PS. PS = 1 causes MS = 1, continuing microprogram execution.
- PS = 0 forces MS = 0, removing the stall, and returning control to the pipeline.
- This causes MI to become 0 (if the new instruction is not also a complex one).
- If CA = 0, the microprogrammed control is locked the IDLE state until MI = 1.

TABLE 10-5
Address Control

Inputs					Outputs			
MZ_{-1}	MZ	MI	PS	Z	ME_1	ME_0	MS	Register Transfer Due to ME
11	01	X	0	0	0	0	1	$\overline{PS} \cdot \overline{Z}$: $MC \leftarrow MC + 1$
11	01	X	0	1	0	1	1	$\overline{PS} \cdot Z$: $MC \leftarrow CA_{-1}$
11	01	X	1	0	0	1	1	$PS \cdot \overline{Z}$: $MC \leftarrow CA_{-1}$
11	01	X	1	1	0	0	0	$PS \cdot Z$: $MC \leftarrow MC + 1$
0X	01	X	X	X	0	0	1	$MC \leftarrow MC + 1$
X0	01	X	X	X	0	0	1	$MC \leftarrow MC + 1$
XX	00	0	X	X	1	0	0	$MC \leftarrow CA$
XX	00	1	X	X	0	1	1	$MC \leftarrow ST$
XX	10	X	0	X	1	0	0	$\overline{PS}$: $MC \leftarrow CA$
XX	10	X	1	X	1	0	1	PS: $MC \leftarrow CA$
XX	11	X	X	X	0	0	1	$MC \leftarrow MC + 1$