Computer Architecture

Single- Cycle Hardwired Control Computer

Dr. Masood Mir

Week 9

Single- Cycle Hardwired Control

- The block diagram for a computer that has a hardwired **control unit** and that fetches and executes an instruction in a **single clock cycle** is shown in Figure.
- The storage resources, instruction formats, and instruction specifications for this computer are same.
- The datapath shown is the same as that in Figure 8-11 with m = 3 and n = 16.
- The **data memory** is attached to the Address out, Data out, and Data in by connections to the datapath.
- A control signal MW is 1 to write the memory, and 0 otherwise.
- The Control unit appears on the left in Figure 8-15.
- The **instruction memory**, together with its address inputs and instruction outputs, is shown for convenience within the control unit.

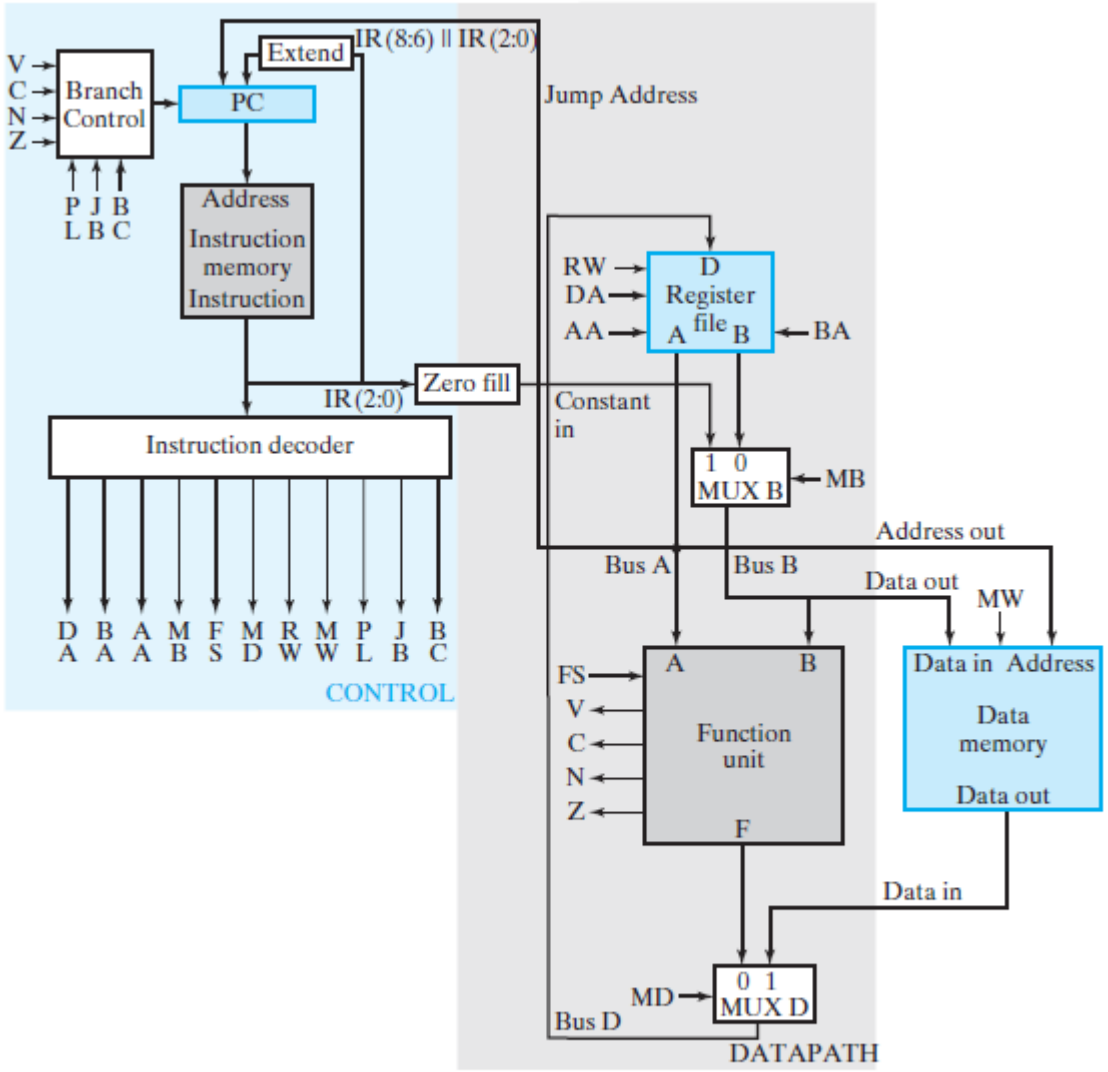


FIGURE 8-15
Block Diagram for a Single-Cycle Computer

Single- Cycle Hardwired Control

- **Instruction memory is not written on during the execution of a program and does not work with Clock**.
- The **PC** provides the instruction address to the instruction memory, and the instruction output from the instruction memory goes to **instruction decoder**.
- The output from the instruction memory also goes to Extend (to provide the address offset to the PC) and Zero fill (to provide the constant input to the datapath).
- **Extend** appends the leftmost bit of the 6-bit address offset field AD to the left of AD, preserving its 2s complement representation.
- **Zero fill** appends 13 zeros to the left of the operand (OP) field of the instruction to form a 16-bit unsigned operand for use in the datapath. For example, operand value 110 becomes 0000000000000110 or +6.

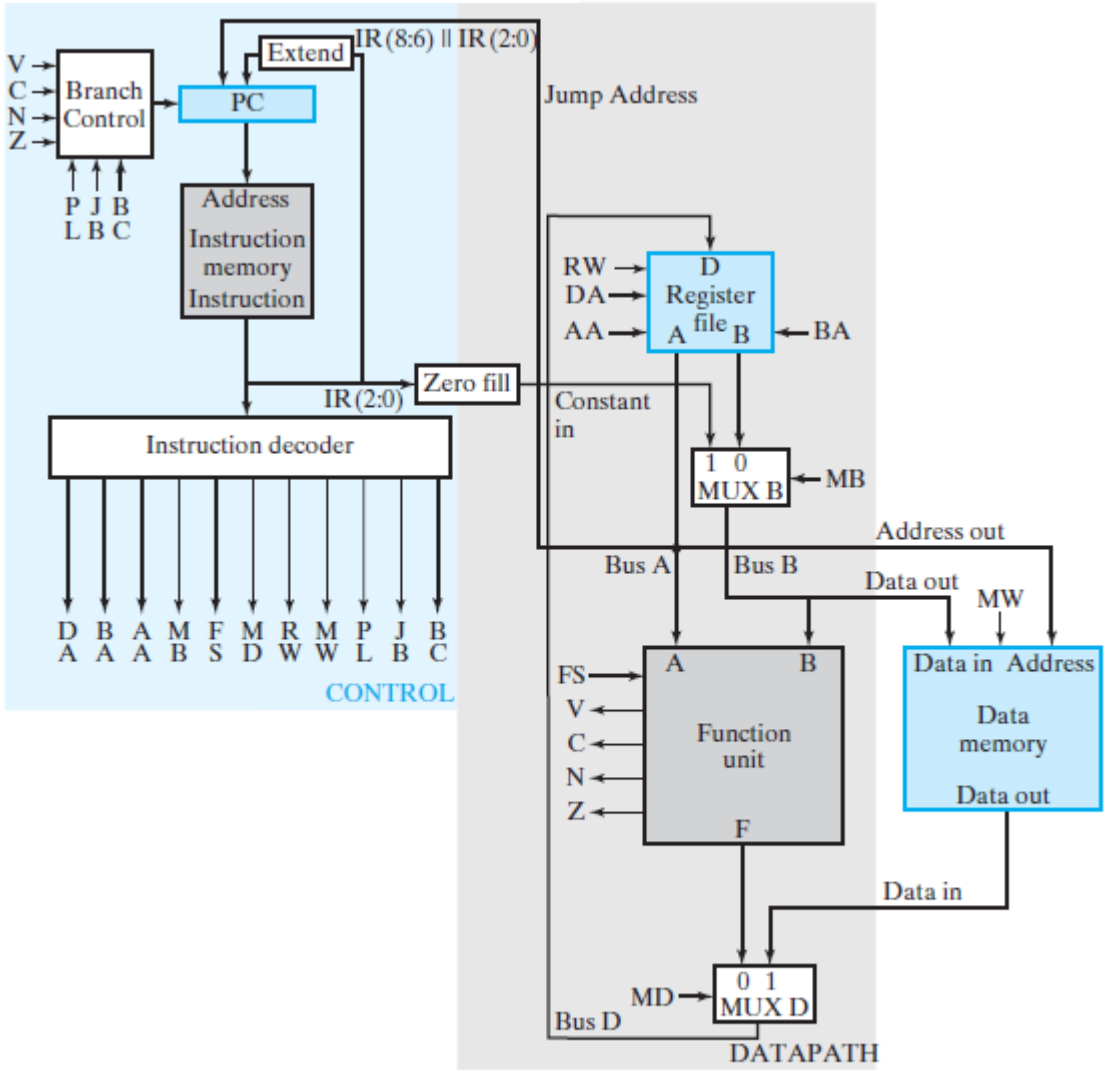


FIGURE 8-15
Block Diagram for a Single-Cycle Computer

Single- Cycle Hardwired Control

- The PC is updated in each clock cycle. The behavior of the PC, which is a complex register, is determined by the **opcode, N, and Z,** since C and V are not used in this control-unit design.
- If a **jump** occurs, the new PC value becomes the value on Bus A.
- If a **branch** is taken, then the new PC value is the sum of the previous PC value and the sign- extended address offset, which in 2s complement can be either positive or negative. Otherwise, the PC is incremented by 1.
- A jump occurs for **bit 13** in the instruction equal to 1. For bit 13 equal to 0, a conditional branch occurs.
- The status bit that is the condition for the branch is selected by bit 9 of the instruction. For **bit 9** equal to 1, N is selected and, for bit 9 equal to 0, Z is selected.

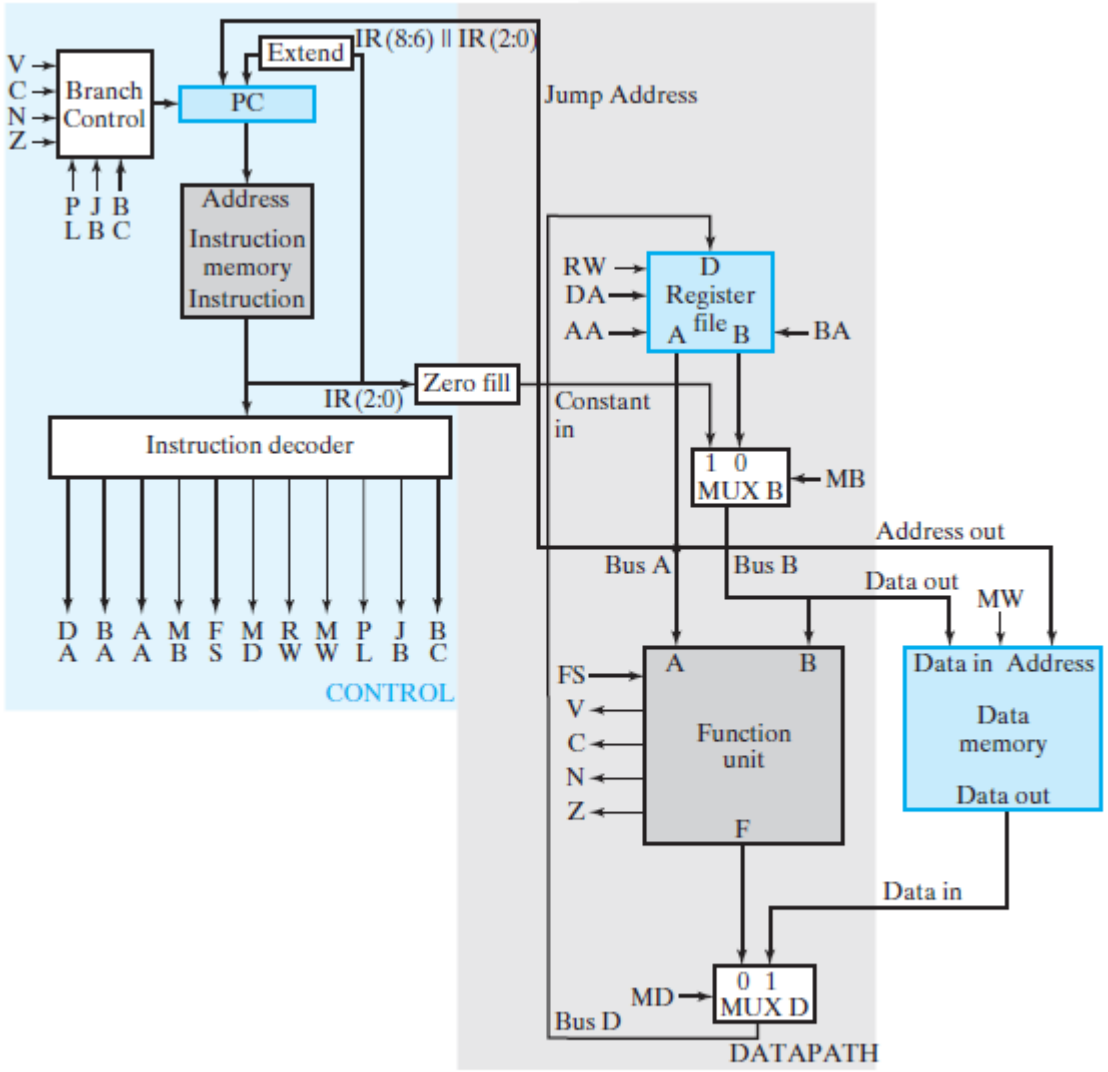


FIGURE 8-15
Block Diagram for a Single-Cycle Computer

Single- Cycle Hardwired Control

- Note that there is no sequential logic in the control part other than the PC.
- Thus, aside from providing the address to the instruction memory, the **control logic is combinational** in this case (Instruction Memory also).
- That fact, combined with the structure of the datapath and the use of separate instruction and data memories, allows the single-cycle computer to obtain and execute an instruction from the instruction memory, all in a single clock cycle.
- The instruction decoder is a combinational circuit that provides all of the control words for the datapath, based on the contents of the fields of the instruction.
- A number of the fields of the control word can be obtained directly from the contents of the fields in the instruction.

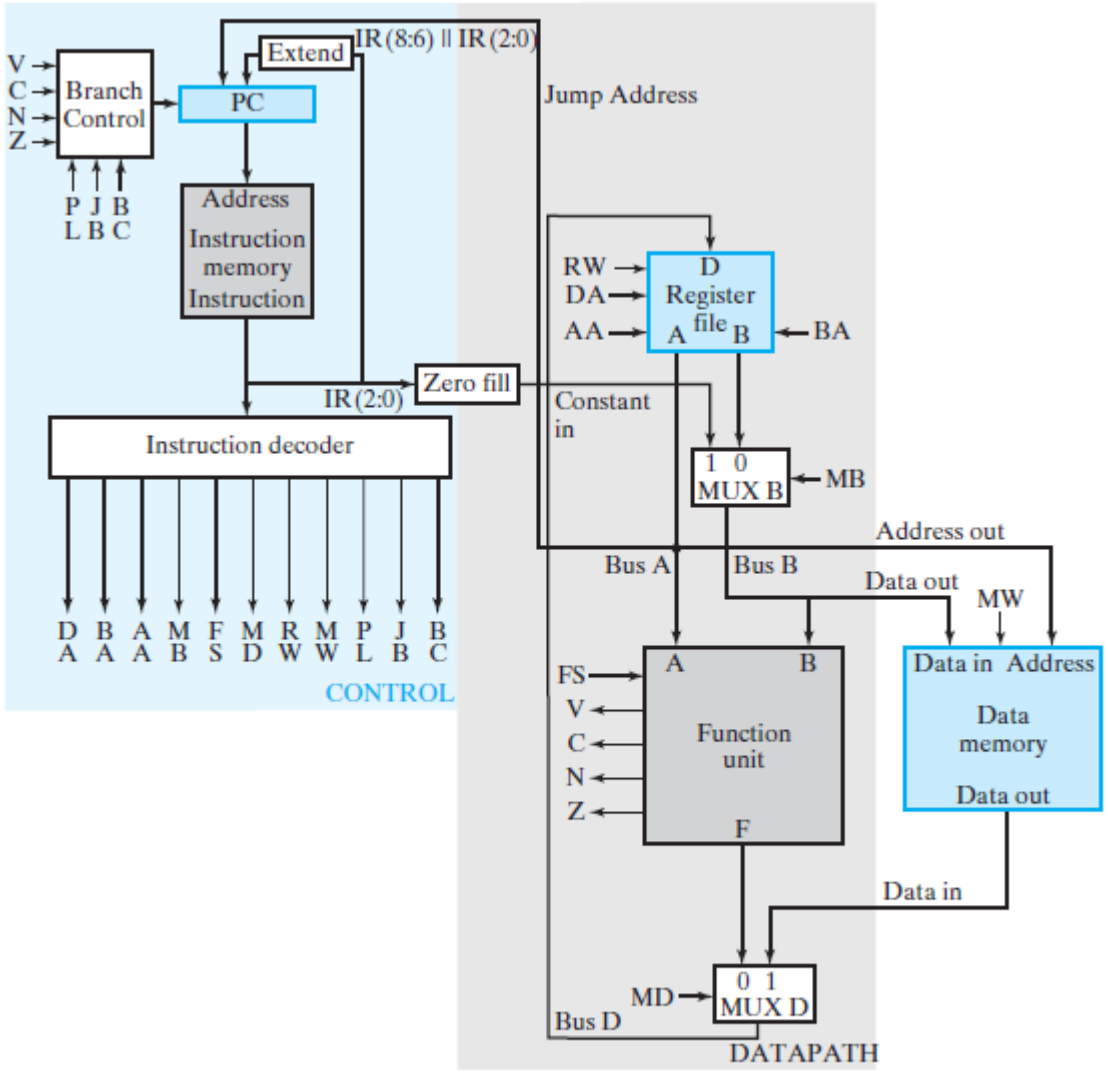


FIGURE 8-15
Block Diagram for a Single-Cycle Computer

Instruction Decoder of Single- Cycle Hardwired Control

- Looking at Figure 8-16, we see that the control-word fields **DA, AA, and BA** are equal to the instruction fields **DR, SA, and SB**, respectively.
- Also, control field **BC** for selection of the branch condition status bits is taken directly from the last bit of Opcode (bit 9 of Instruction).
- The remaining control-word fields include datapath and data memory control bits **MB, MD, RW, and MW**.
- MB is bit 15 of Instruction, MD is bit 13, RW is $\overline{bit14}$, bit 14, MW is bit14 . $\overline{bit15}$
- There are two added bits for the control of the PC: **PL** and **JB**. If there is to be a jump or branch, PL = 1, loading the PC. For PL = 0, the PC is incremented.
- With PL = 1, JB = 1 calls for a jump, and JB = 0 calls for a conditional branch.

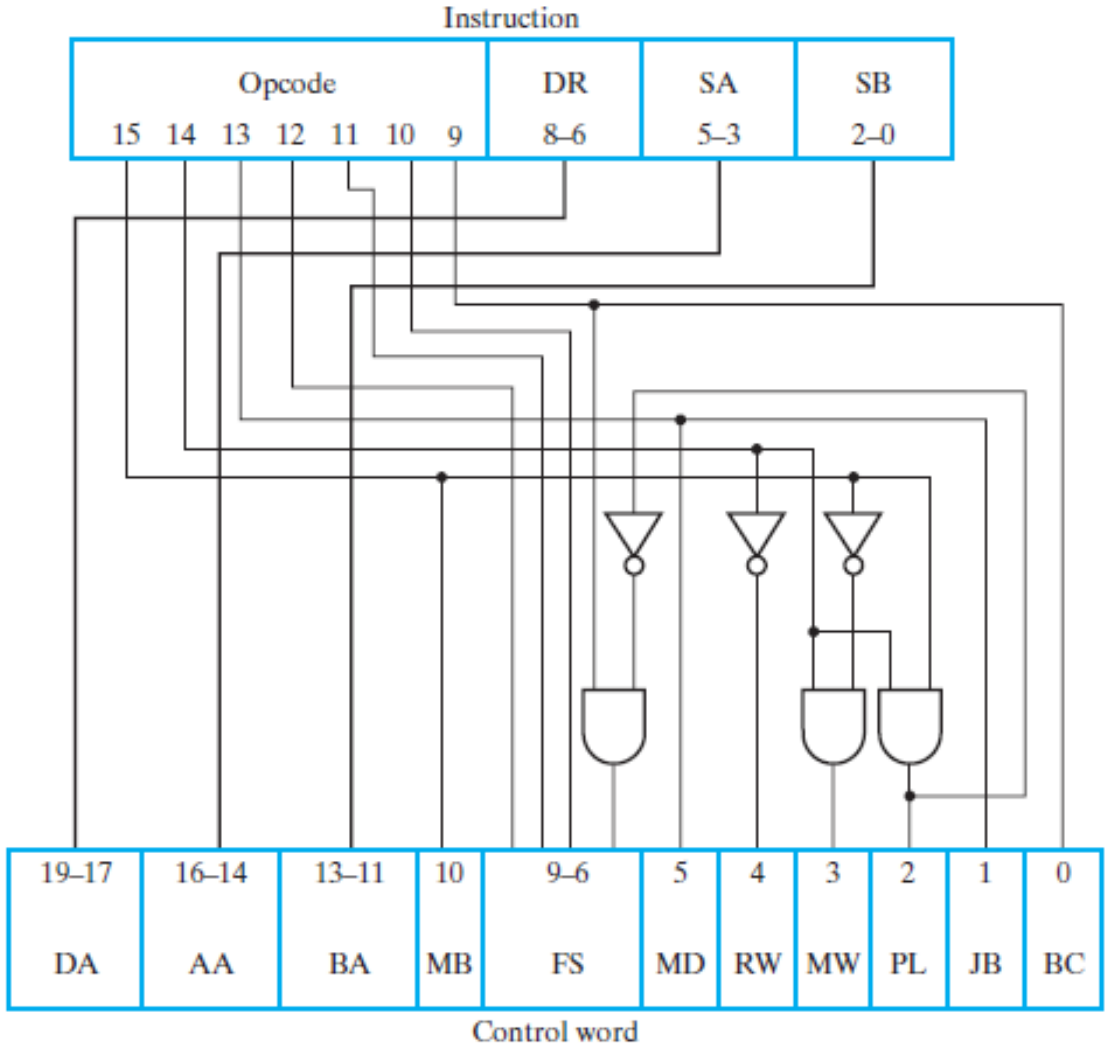


FIGURE 8-16
Diagram of Instruction Decoder

Instruction Decoder of Single- Cycle Hardwired Control

- We divide the various instructions possible for the simple computer into different **function types** and then assign the **three bits** of the opcode to the various types.
- The instruction function types shown in Table 8-10 are based on the **use of specific hardware resources** in the computer, such as MUX B, the Function unit, the Register file, Data memory, and the PC.
- For example, the first function type uses the ALU, sets MUX B to use the Register file source, sets MUX D to use the Function unit output, and writes to the Register file.
- Other instruction function types are defined as various combinations of use of a constant input instead of a register, Data memory reads and writes, and manipulation of the PC for jumps and branches.

TABLE 8-10
Truth Table for Instruction Decoder Logic

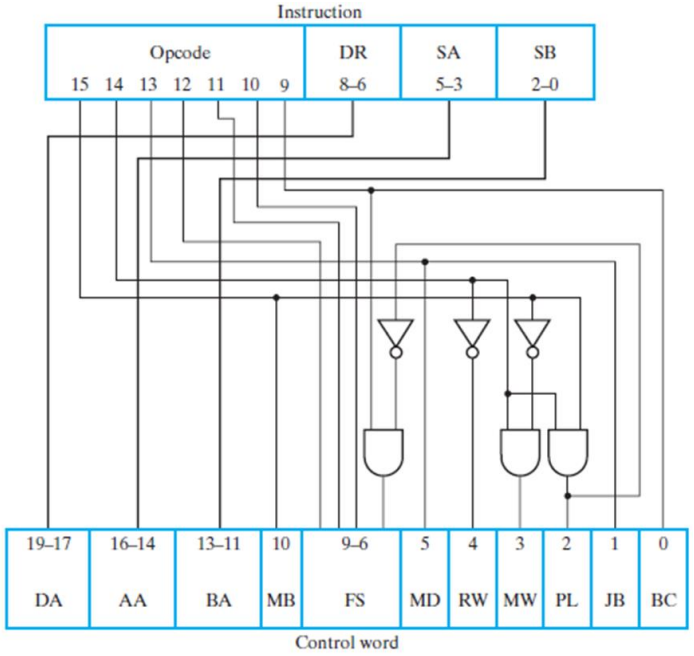
	Instruction Bits				Control-Word Bits						
Instruction Function Type	15	14	13	9	MB	MD	RW	MW	PL	JB	BC
Function-unit operations using registers	0	0	0	X	0	0	1	0	0	X	X
Memory read	0	0	1	X	0	1	1	0	0	X	X
Memory write	0	1	0	X	0	X	0	1	0	X	X
Function-unit operations using register and constant	1	0	0	X	1	0	1	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

Assignment: Look at the instruction set and specify which instruction is in which function type.

Instruction Decoder of Single- Cycle Hardwired Control

TABLE 8-10
Truth Table for Instruction Decoder Logic

	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



- By looking at the relationship between the instruction function types and the necessary control-word values needed for their implementation, bits 15 through 13 and bit 9 were assigned as shown in Table 8-10. This assignment attempted to minimize the logic required to implement the decoder.
- To perform the design of the decoder, the values for all of the single-bit fields in the control word were determined from the function types and entered into Table 8-10.
- Note that there are a number of don’t-care (X) entries. Treating Table 8-10 as a truth table and optimizing the logic functions, the logic for the single-bit outputs of the instruction decoder in Figure 8-16 results.

Instruction Decoder of Single- Cycle Hardwired Control

- In the optimization, the unused codes for bits 15, 14, 13, and 9 were assumed to have X values for all of the single bit fields. Not to be appeared in a program.
- A more conservative design specifies **RW, MW, and PL all zero for these unused codes** to insure that the storage resource state is unchanged for these unused codes.
- The optimization results in the logic in Figure 8-16 for implementing MB, MD, RW, MW, PL, and JB.

TABLE 8-10
Truth Table for Instruction Decoder Logic

	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

- The **remaining logic** in the decoder deals with the **FS field**. For all but the conditional branch and unconditional jump instructions, bits 9 through 12 are fed directly through to form the FS field.
- During **conditional branch** operations, such as Branch on Zero, the value in **source register A** must be passed through the ALU so that the status bits **N** and **Z** can be **evaluated**. This requires **FS = 0000**. The use of bit 9, however, for status-bit selection for conditional branches is required.
- We add an enable on bit 9 that forces FS0 to zero whenever PL = 1, as shown in Figure 8-16, to avoid contradiction in values between bit 9 and FS.

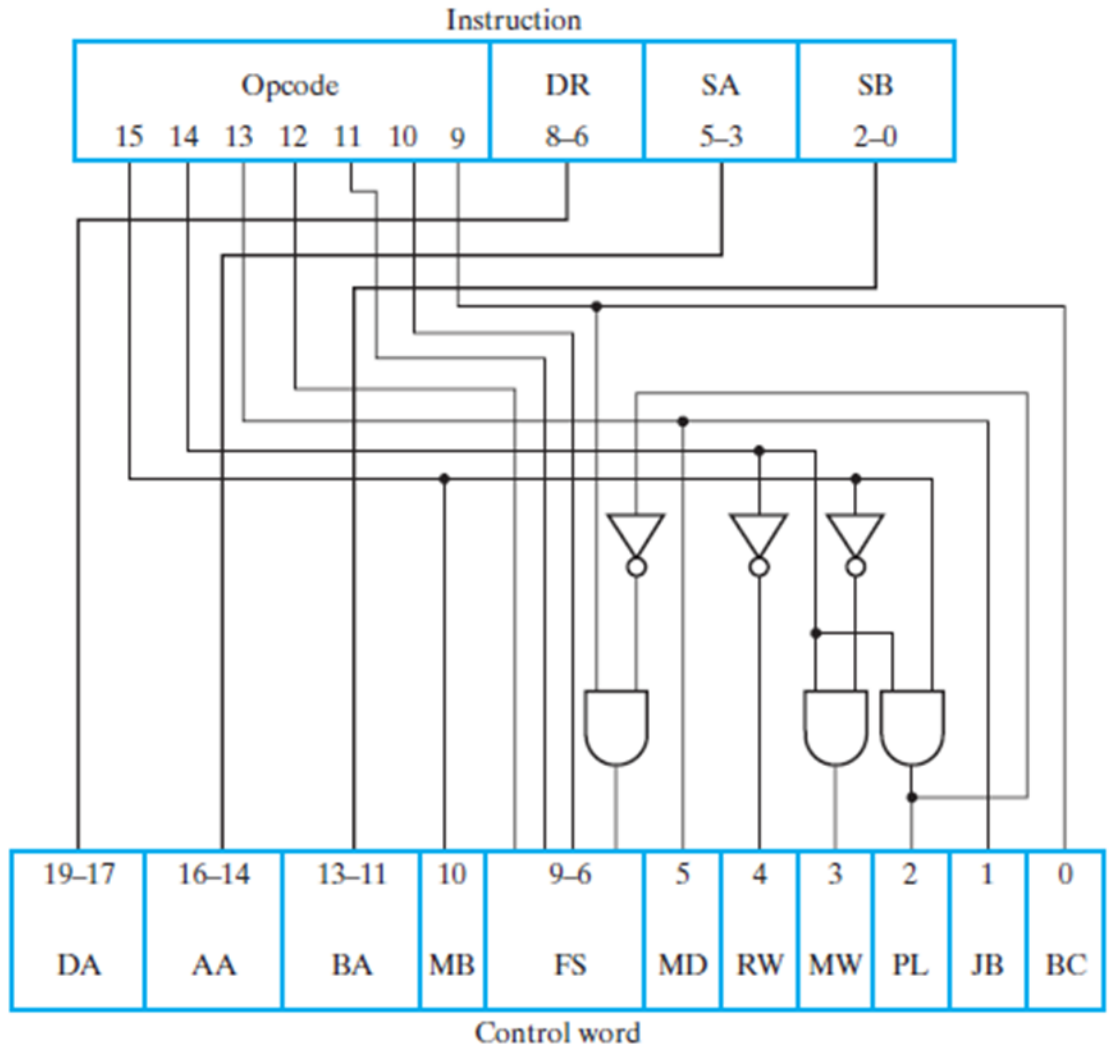


FIGURE 8-16
Diagram of Instruction Decoder