Sequencing and Control

Instructor: Saraju P. Mohanty

PART 2

- Microprogrammed Control
- A Simple Computer Architecture
- Single-Cycles Hardwired Control
- Pipelined Control

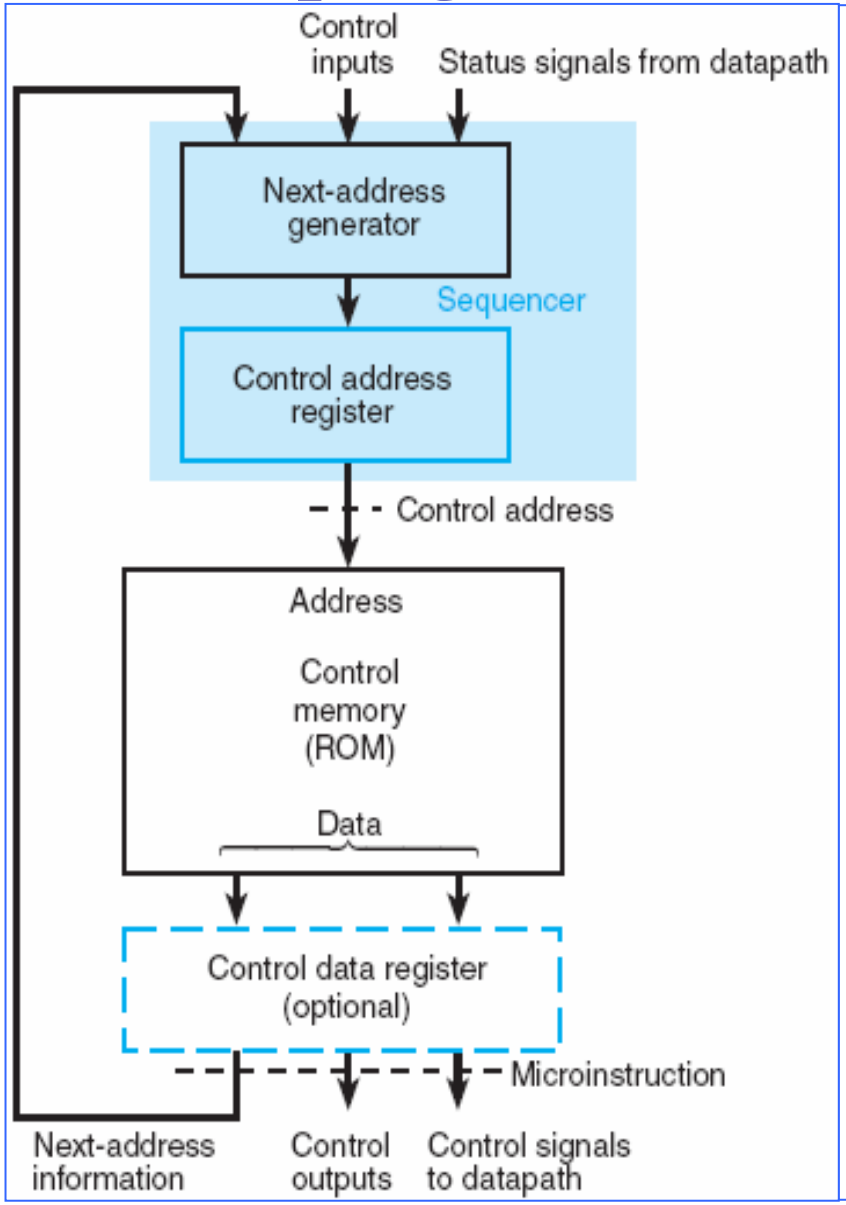
Sources

- Logic and Computer Design Fundamentals by M. M. Mano and C. R. Kime.
- Dr. Valavanis lectures

Microprogrammed Control

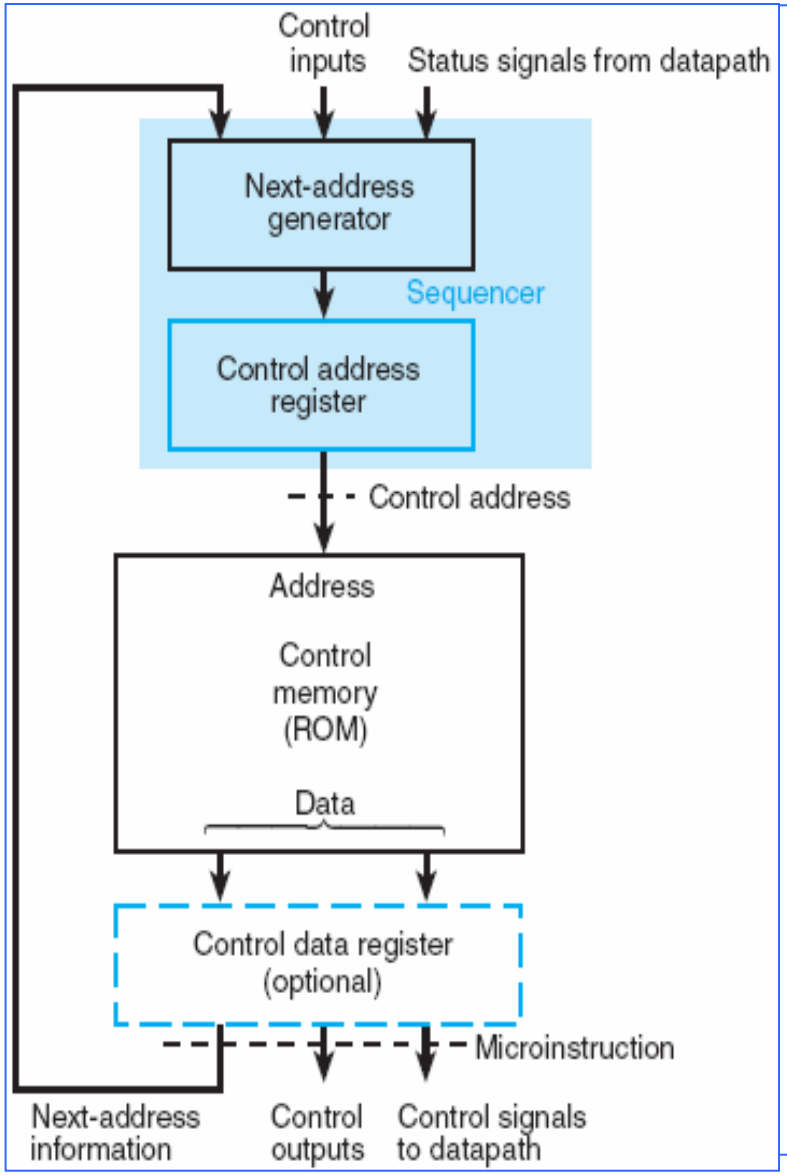
- Microprogrammed control – A control unit with its binary control values stored as words in memory.
  - Each word is a microinstruction, which specifies one or more microoperations for the system.
  - Microprogram – A sequence of microinstructions.
- Microprogram is fixed, so it is often stored in a ROM.
- Microprogram can also be stored in a RAM, but has to be loaded initially at system startup from the computer console or from a nonvolatile storage (like magnetic disk).
- With either RAM or ROM, the memory in the control unit is called Control memory.
- If RAM is used the memory is called writable control memory.

Microprogrammed Control Unit Organization



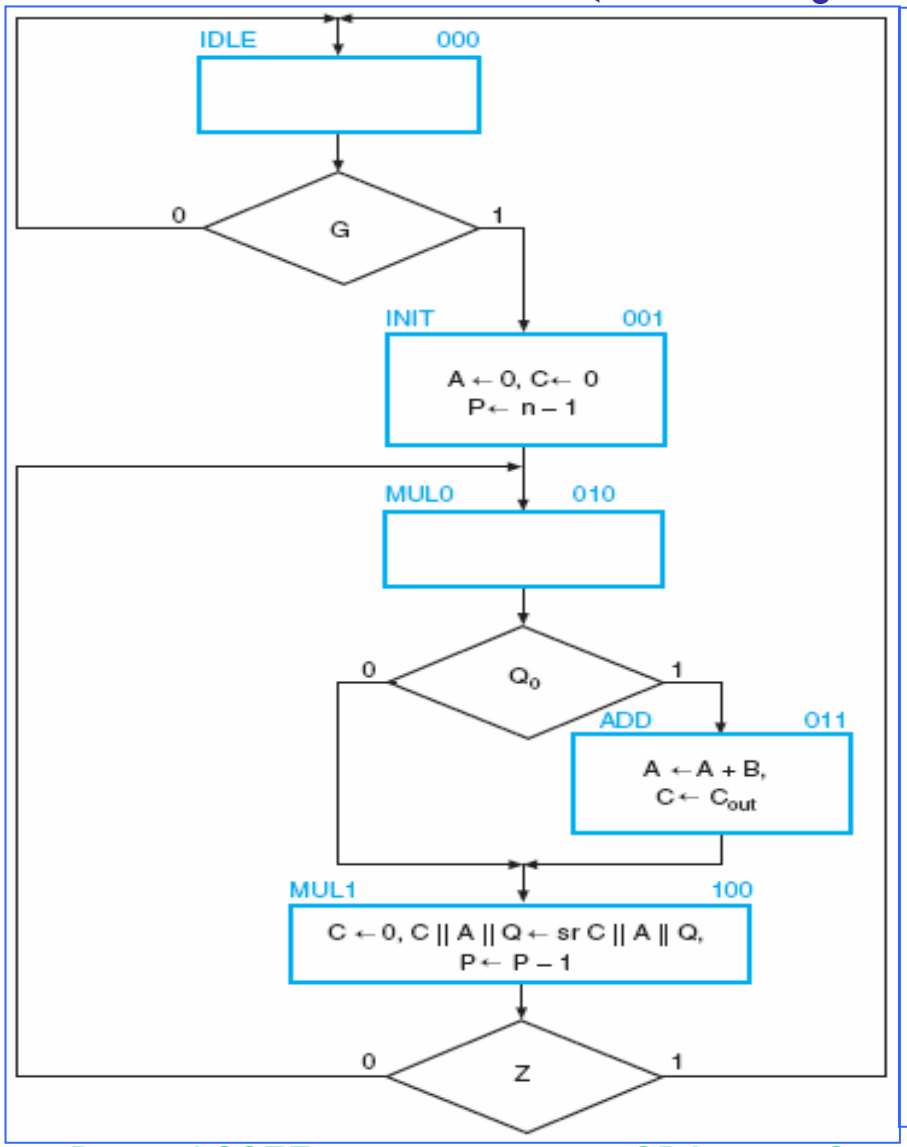
- The control memory assumed to be a ROM within which all control information is permanently stored.
- Control Address Register (CAR) – specifies the address of the microinstruction.
- Control Data Register (CDR) (optional) – May hold the microinstruction currently being executed by the datapath and control unit.
- Next Address Generator produces and transfers the next address to the CAR.

Microprogrammed Control Unit ……..



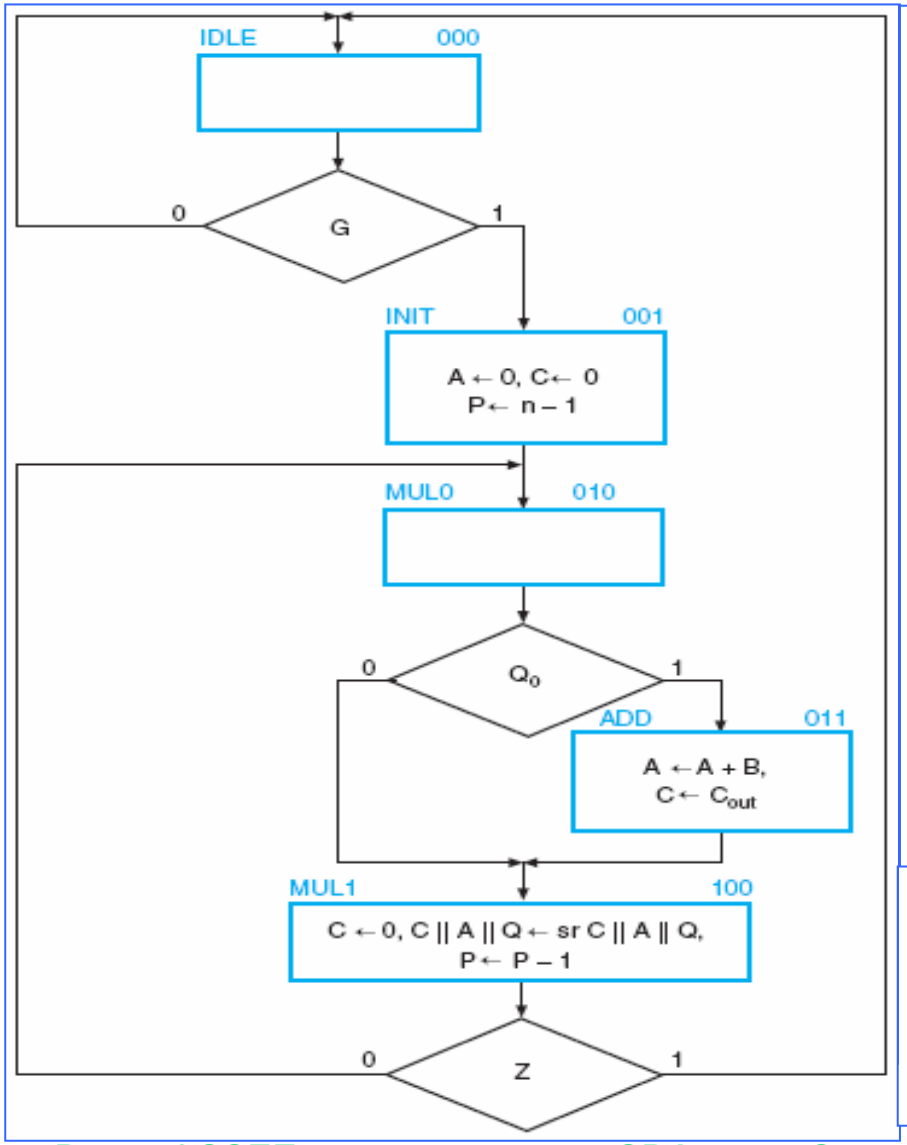
- The next-address generator in combination with the CAR is called Sequencer– determines the sequence of instruction that is read from the control memory.
- Typical function of a microprogram sequencer is incrementing the CAR by one and loading the CAR.
- The ROM operates as a combinational circuit, with the addresses as the input and the corresponding microinstruction as the output.

Microprogrammed Control Unit (Binary Multiplier)



- The circuit depicted is a Moore-type circuit, hence conditional output boxes are not permitted in the ASM chart.
- Compared to the original ASM chart, this chart has two more states, such as INIT and ADD.
- The ASM has only single decision boxes determining the sequencing between states.
- Next-state decisions based on multiple values are excluded in simpler next-address generator designs.

Microprogrammed Control Unit
(Binary Multiplier ….)



- We need to determine three things:
  - the bits in the control word for microinstruction
  - the sizes of the ROM and the CAR
  - the structure of the next address generator
- Then we design the microsequencer and write the microprogram for binary multiplier.

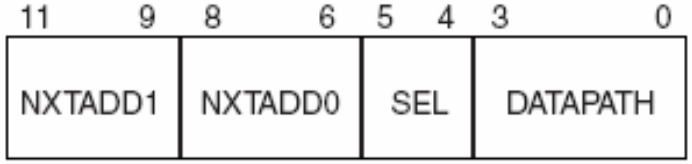


Fig. 8-18 Microinstruction Control Word Format

Microprogrammed Control Unit
(Binary Multiplier ….)

Block Diagram Module	Microoperation	Control Signal Name	Control Expression
Register A:	$A \leftarrow 0$	Initialize	$\text{IDLE} \cdot G$
	$A \leftarrow A + B$	Load	$\text{MUL0} \cdot Q_0$
	$C \| A \| Q \leftarrow \text{sr}\ C \| A \| Q$	Shift_dec	MUL1
Register B:	$B \leftarrow IN$	Load_B	LOADB
Flip-Flop C:	$C \leftarrow 0$	Clear_C	$\text{IDLE} \cdot G + \text{MUL1}$
	$C \leftarrow C_{\text{out}}$	Load	—
Register Q:	$Q \leftarrow IN$	Load_Q	LOADQ
	$C \| A \| Q \leftarrow \text{sr}\ C \| A \| Q$	Shift_dec	—
Counter P:	$P \leftarrow n - 1$	Initialize	—
	$P \leftarrow P - 1$	Shift_dec	—

Table 8-1 Control Signals for Binary Multiplier

Only 4 control signals needed for the datapath to perform multiplication: Initialize, Load, Clear_C, Shift_dec.

Microprogrammed Control Unit
(Binary Multiplier ….)

Control Signal	Register Transfers	States in Which Signal is Active	Micro-instruction Bit Position	Symbolic Notation
Initialize	$A \leftarrow 0, P \leftarrow n-1$	INIT	0	IT
Load	$A \leftarrow A+B, C \leftarrow C_{out}$	ADD	1	LD
Clear_C	$C \leftarrow 0$	INIT, MUL1	2	CC
Shift_dec	$C\|\|A\|\|Q \leftarrow \text{sr } C\|\|A\|\|Q, P \leftarrow P-1$	MUL1	3	SD

Table 8-3 Control Signals for Microprogrammed Multiplier Control

For the above table, the four control signal combinations used are:

- Initialize and Clear_C in state INIT
- Load in state ADD
- Clear_C and Shift_dec in state MUL1
- No signal in state IDLE and state MUL0

Microprogrammed Control Unit
(Binary Multiplier ….)

- The 4-bit codes from the control signals table are used in the DATAPATH field of the word.
- It is possible to manage with 2 bits along with a decoder, but for small circuits reducing ROM size and adding decoder is not justified.
- The remainder of the microinstruction control word is devoted to the sequencing of the control unit.
- There are many ways to design the sequencer, which determines the fields needed for microsequencing.
- The preliminary step, sequencing requirement defined by ASM chart, i.e. how next state is reached from a present state.
- The approach used to define address is another important step in designing a sequencer.

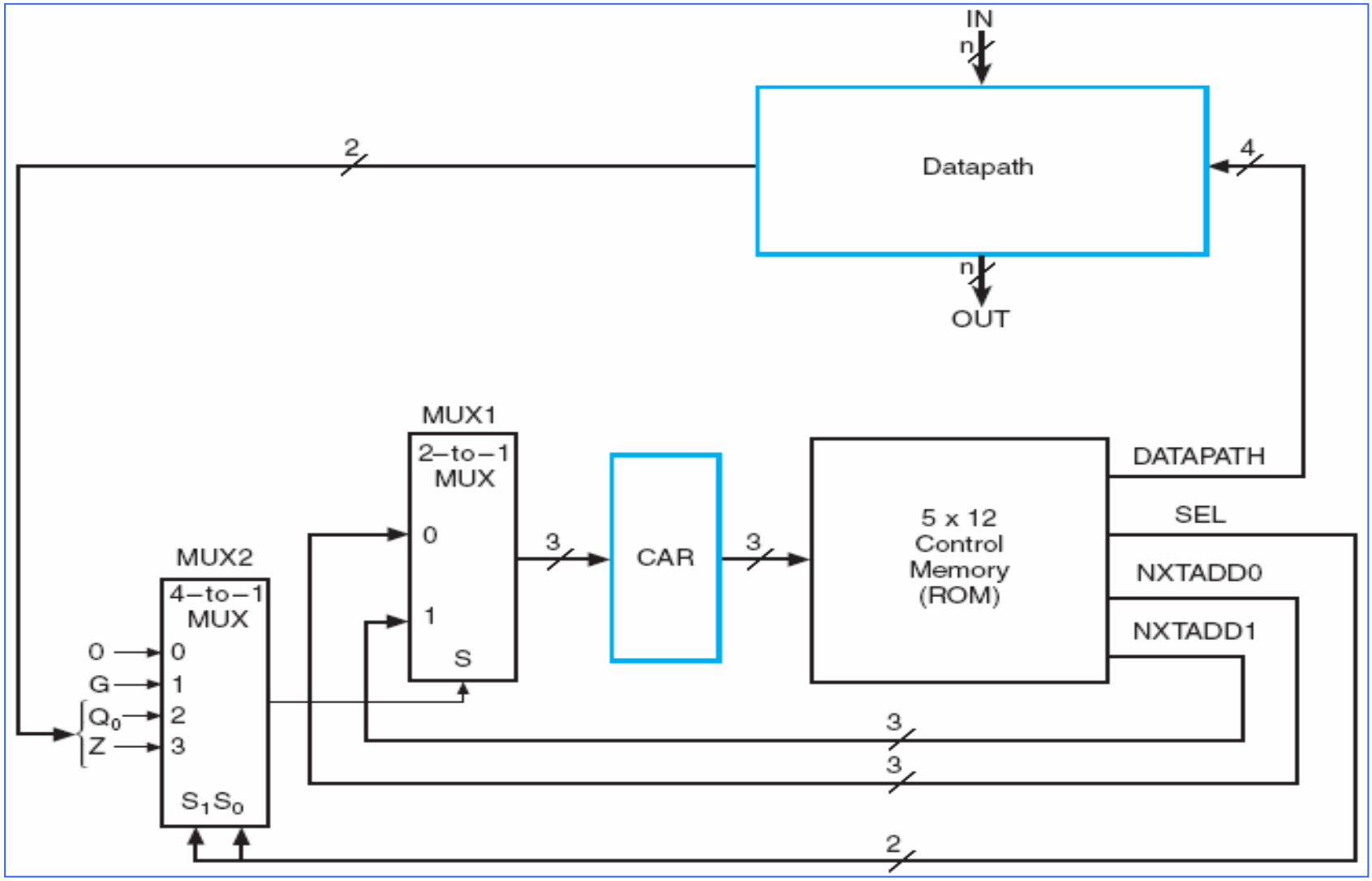
Microprogrammed Control Unit (Binary Multiplier ….)

SEL		
Symbolic notation	**Binary Code**	**Sequencing Microoperations**
NXT	00	$CAR \leftarrow NXTADD0$
DG	01	$\overline{G}$: $CAR \leftarrow NXTADD0$ G: $CAR \leftarrow NXTADD1$
DQ	10	$\overline{Q_0}$: $CAR \leftarrow NXTADD0$ Q_0: $CAR \leftarrow NXTADD1$
DZ	11	$\overline{Z}$: $CAR \leftarrow NXTADD0$ Z: $CAR \leftarrow NXTADD1$

Table 8-4 SEL Field Definition for Binary Multiplier Control Sequencing

NOTE: In RTL, If (K=1) then (R2 ← R1) is written as K: R2 ← R1

Microprogrammed Control Unit
(Binary Multiplier ….)



Microprogrammed Control Unit
(Binary Multiplier ….)

- The length of the ROM control words is 12 bits.
- Since there are only 5 states in the ACM chart, the ROM contains 5 words.
- 3 bits are used to address the 5 words and a simple 3-bit parallel load register is used the CAR.
- The addresses to be loaded into the CAR come from the NXTADD0 and NXTADD1 in the microinstruction on the ROM output.
- A 2-to-1 multiplexer selects between two address sources.
- A 4-to-1 multiplexer is used to select constant or decision variable required for each of the four SEL codes.

Microprogrammed Control Unit (Binary Multiplier ….)

Register Transfer Description of Binary Multiplier Microprogram

Address	Symbolic transfer statement
IDLE	G: $CAR \leftarrow$ INIT, $\overline{G}$: $CAR \leftarrow$ IDLE
INIT	$C \leftarrow 0, A \leftarrow 0, P \leftarrow n-1, CAR \leftarrow$ MUL0
MUL0	Q_0: $CAR \leftarrow$ ADD, $\overline{Q_0}$: $CAR \leftarrow$ MUL1
ADD	$A \leftarrow A + B, C \leftarrow C_{out}, CAR \leftarrow$ MUL1
MUL1	$C \leftarrow 0, C \Vert A \Vert Q \leftarrow \text{sr}\ C \Vert A \Vert Q$, Z: $CAR \leftarrow$ IDLE, $\overline{Z}$: $CAR \leftarrow$ MUL0, $P \leftarrow P-1$

Symbolic Microprogram and Binary Microprogram for Multiplier

Address	NXTADD1	NXTADD0	SEL	DATAPATH	Address	NXTADD1	NXTADD0	SEL	DATAPATH
IDLE	INIT	IDLE	DG	None	000	001	000	01	0000
INIT	—	MUL0	NXT	IT, CC	001	000	010	00	0101
MUL0	ADD	MUL1	DQ	None	010	011	100	10	0000
ADD	—	MUL1	NXT	LD	011	000	100	00	0010
MUL1	IDLE	MUL0	DZ	CC, SD	100	000	010	11	1100

A Simple Computer Architecture

- Program – A list of instructions that specifies the operations, the operands, and the sequence in which processing is to occur.
- Data processing performed by a computer can be altered by specifying a new program with different instructions or by specifying the same instructions with different data.
- Instruction – is a collection of bits that instructs the computer to perform a specific operation.
- Operation code (opcode) – A group of bits in the instruction that specifies an operation, such as add, subtract, shift, or complement.
- The collection of instructions for a computer is called its instruction set and a thorough description of the instruction set is called its instruction set architecture.
- An operand is said to be specified explicitly is the instructions contain a special bit for it identification, e.g. in addition operation, whereas operand may specified implicitly as a part of definition of operation itself as in the case on increment register operation +1 is implicit.

A Simple Computer Architecture (Instruction Formats)

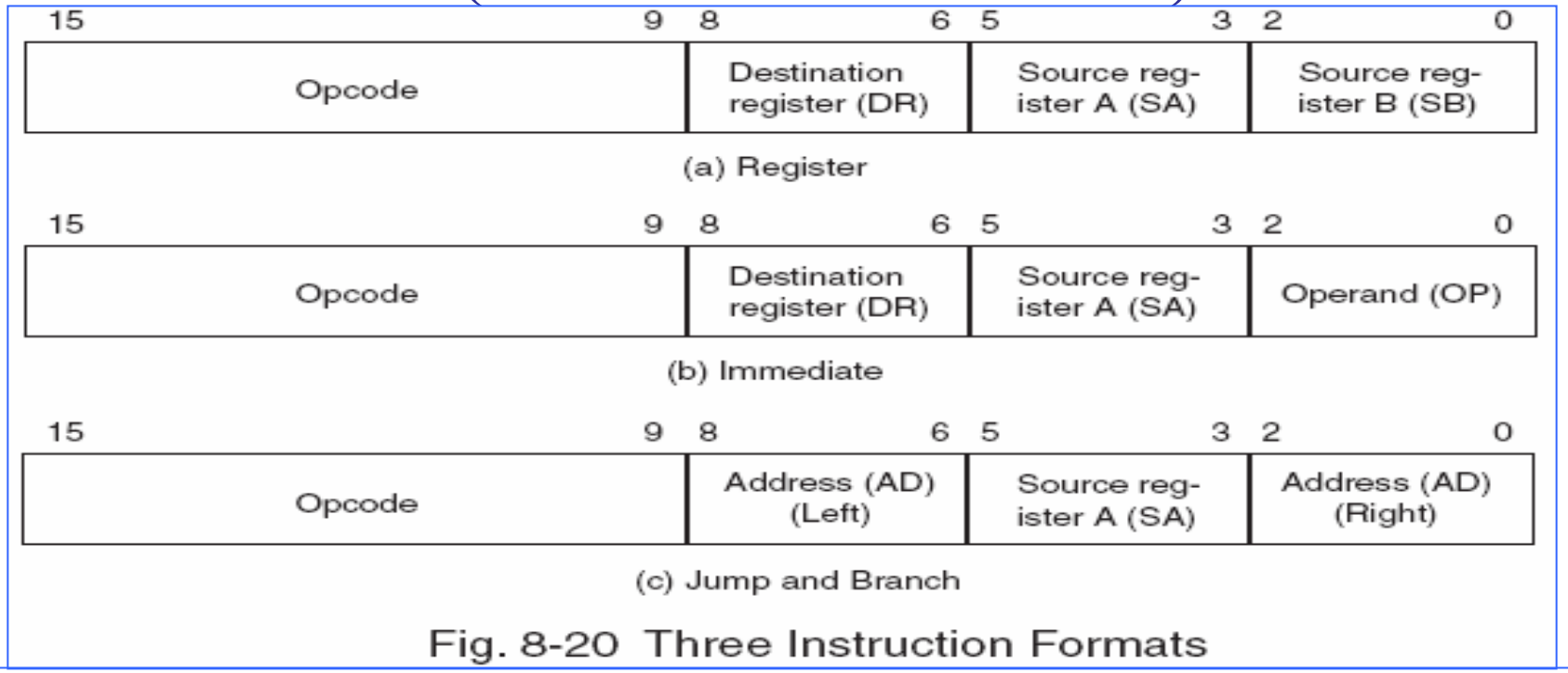


Fig. 8-20 Three Instruction Formats

- The instruction format is shown as a rectangular box which shows the bits of the instruction, as they appear in memory words or in a control register.
- The bits of the instruction are divided into groups or parts called fields.
- Each field, associated with a specific item, specifies a different function for the instruction and all the fields together constitute the instruction format.

A Simple Computer Architecture
Instructions and Data in memory

Decimal address	Memory contents	Decimal opcode	Other specified 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 Immedi-ate)	DR:2 SA:7 OP:3	R2 ← R7 + 3
55	1100011 101 110 100	96 (Branch on zero)	AD: 44 SA:6	If R6 = 0, PC ← PC − 20
70	0000000 011 000 000	Data = 192. After execution of instruction in 35, Data = 80.		

- The instructions and data are placed in memory in binary form as shown above.
- Instructions of the four formats are shown in the figure.

A Simple Computer Arch: Storage Resources

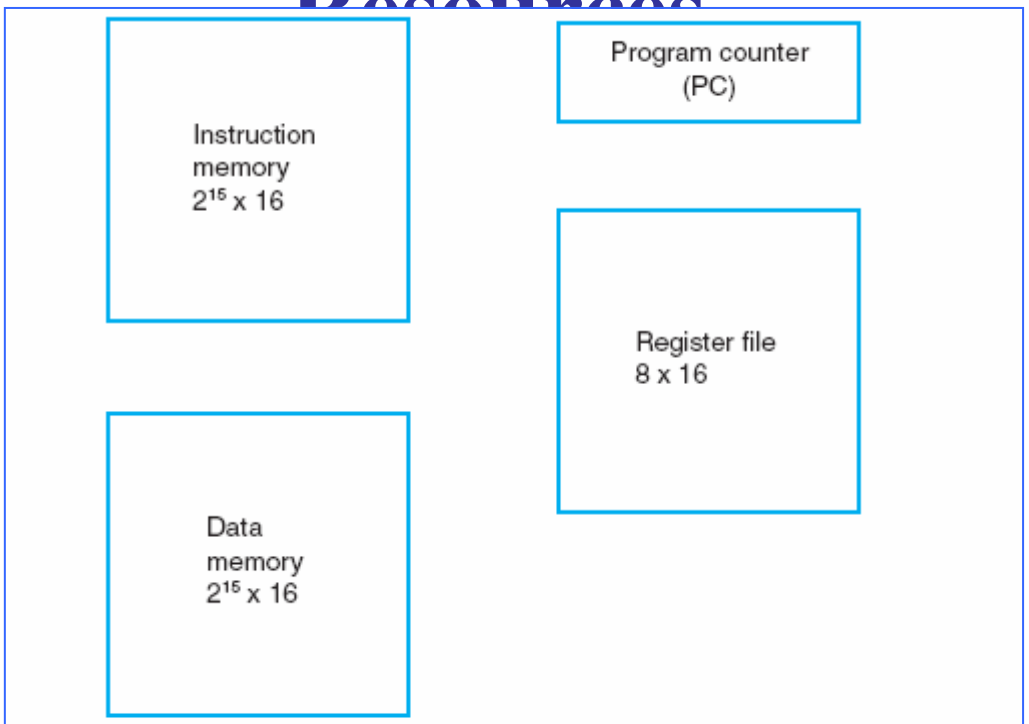


Fig. 8-22 Storage Resource Diagram for a Simple Computer

- The architecture includes two memories, one for storage of instructions and the other for storage of data.
- There can be two different memories for instructions and data, or, they could be in the same memory, but viewed as different from the standpoint of the CPU.
- The architecture also contains a register file with eight 16-bit registers and a 16-bit program counter.

Single-Cycle Hardwired Control

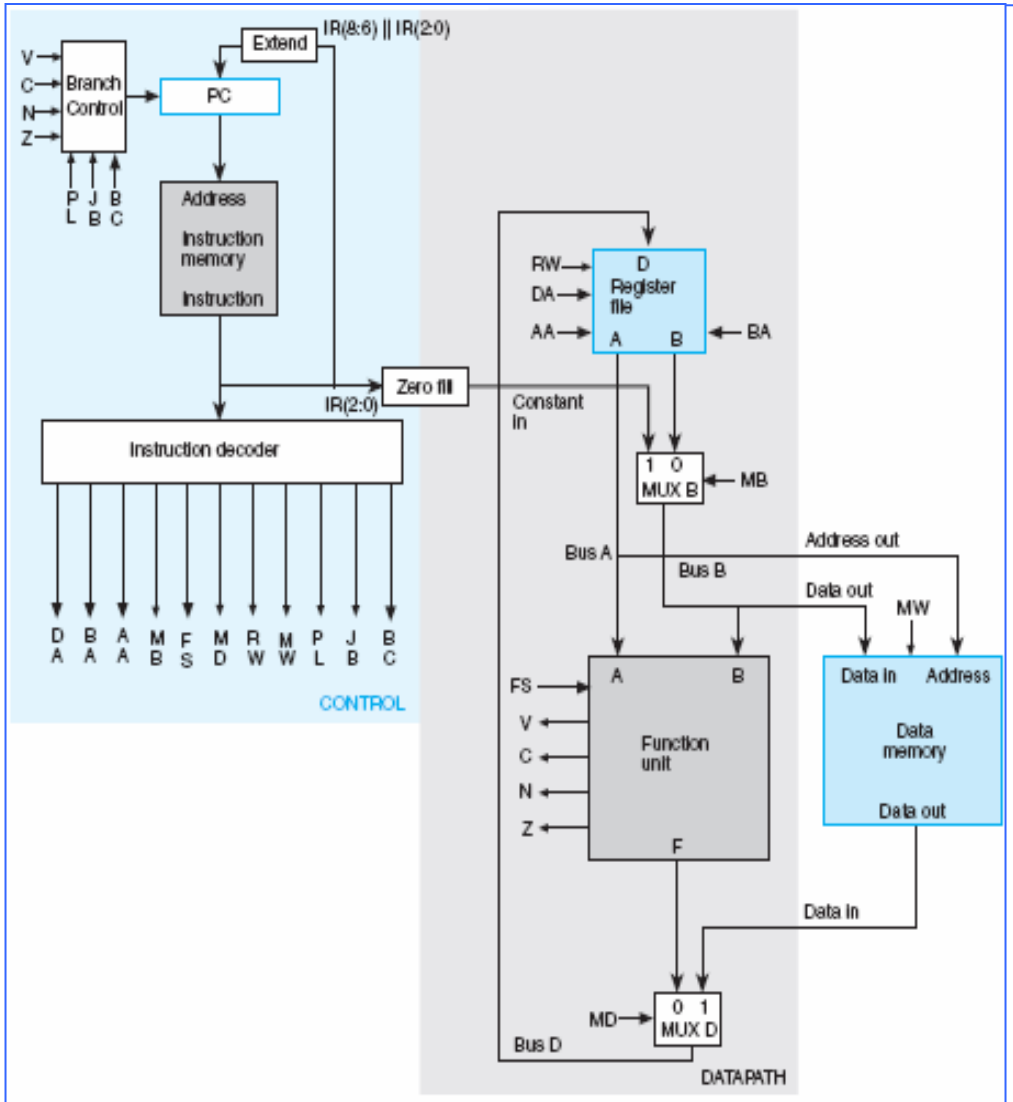


Fig. 8-23 Block Diagram for a Single-Cycle Computer

- The computer fetches and executes an instruction in a single clock cycle.
- The memory is viewed as a combinational rather than a sequential component.
- PC provides the instruction address to instruction memory.
- Instruction goes from instruction-memory to instruction-decoder in control logic.
- The PC is updated at the beginning of each clock cycle.
- If a jump occurs or a branch is taken in the prior clock cycle, then the new PC value is the sum of the previous PC value and the sign-extended address offset, otherwise, the PC is incremented by 1.

Single-Cycle Hardwired Control
Instruction Decoder

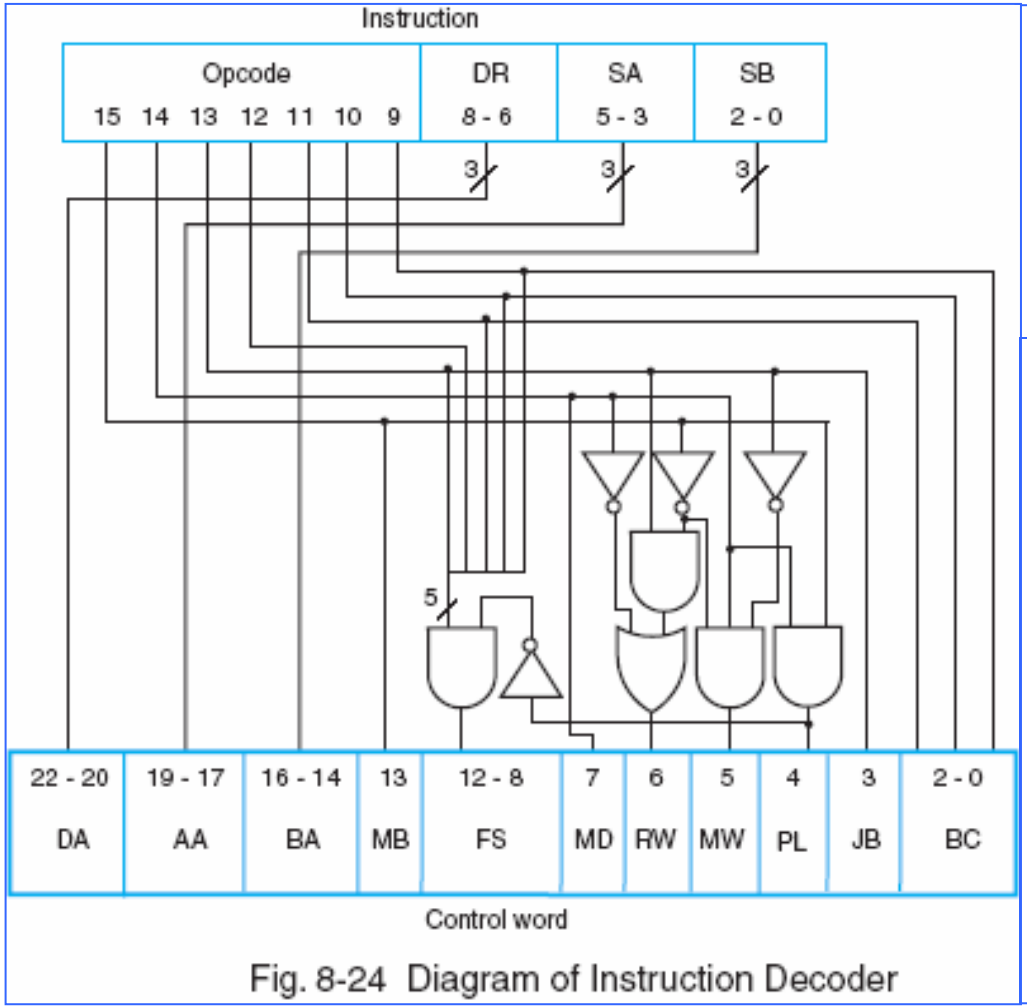


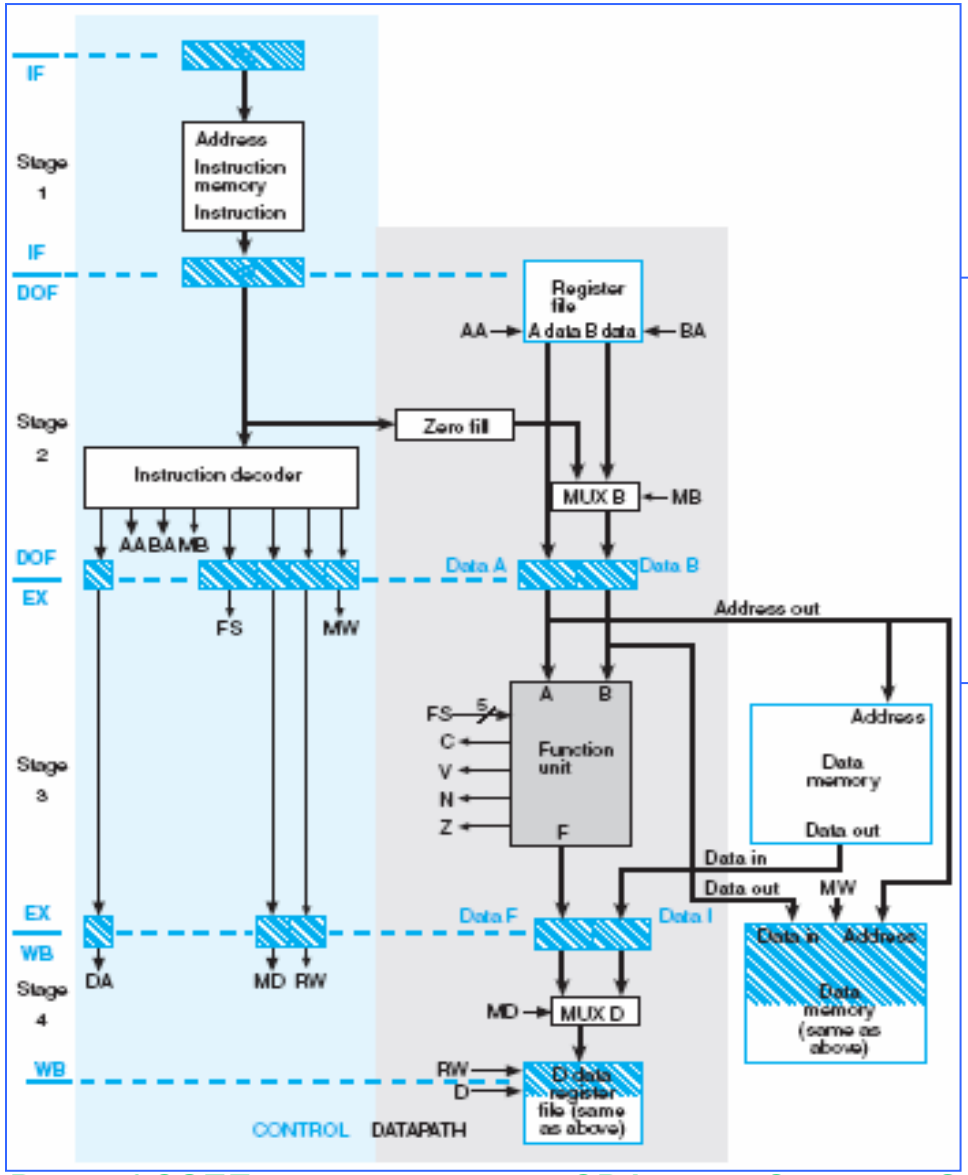
Fig. 8-24 Diagram of Instruction Decoder

The Instruction decoder is a combinational circuit that provides all the control words for the datapath, based on the contents of the fields of the instruction.

Single-Cycle computer - Shortcomings

- Many complex operations cannot be accomplished
- For computers with instructions and data in the same memory, cannot be implemented with a single-cycle system.
- The single-cycle computer has a lower limit on the clock period based on a long worst case delay path.

Pipelined computer



Registers have been added to the pipeline platforms between stages, to pass the decoded instruction information through the pipeline along with the data being processed.