Microprogrammed Control

By,
Er. Nabaraj Bahadur Negi

Contents

- **4.1 Introduction:** Hardwired and Microprogrammed Control Unit, Control Word, Microprogram, Control Memory, Control Address Register, Sequencer,
- **4.2 Address Sequencing:** Conditional Branch, Mapping of Instructions, Subroutines, Microinstruction Format, Symbolic Microinstructions
- **4.3 Design of Control Unit**: Decoding, Microprogram Sequencer.

Control Unit

- Control unit generates timing and control signals for the operations of the computer.
- It's the part of the CPU that initiates sequences of microoperations.
- It tells the computers memory ,arithmetic and logic unit and I/O devices how to respond to a programs instructions.
- Two methods to implement the control unit:
- **Hardwired Control Unit:(**uses fixes instructions, combinational logic units of AND/OR(logic gates),encoders, decoders, etc.)
- **Microprogrammed Control Unit:(**the logic of the control unit is specified by microprograms (consists of a sequence of instructions that specify microoperations).

Note: refer unit-3 for more details.

Terminology

Microprogram

- Program stored in memory that generates all the control signals required to execute the instruction set correctly.
- Consists of microinstructions

Microinstruction

- Contains a control word and a sequencing word
    - **Control Word** - All the control information required for one clock cycle. Or
        - The control variables at any time are represented by 1's and 0's knows as control word.
        - Control words can be programmed to perform various operations.
        - **Sequencing Word** -Information needed to decide the next microinstruction address.

Microoperations

- In computer central processing units, micro-operations (also known as a micro-ops or μops) are detailed low-level instructions used in some designs to implement complex machine instructions (sometimes termed macro-instructions in this context).

Microcode

- Microinstructions can be saved by employing subroutines that use common sections of microcode.
- For example, the sequence of micro operations needed to generate the effective address of the operand for an instruction is common to all memory reference instructions.
- This sequence could be a subroutine that is called from within many other routines to execute the effective address computation.

Control memory (control storage: CS)

- Storage in the microprogrammed control unit to store the microprogram.
- Each word in control memory contains within it a microinstruction.

Writeable control memory (writeable control storage: WCS)

- Control Storage whose contents can be modified, allow the change in microprogram and Instruction set can be changed or modified is referred as Writeable Control Memory**.**

Dynamic microprogramming

- Permits a microprogram to be loaded initially from an auxiliary memory such as a magnetic disk.
- Computer system whose control unit is implemented with a microprogram in WCS.

  - Microprogram can be changed by a systems programmer or a user, though it is mostly used for reading.

Sequencer (Microprogram Sequencer)

- A Microprogram Control Unit that determines the Microinstruction Address to be executed in the next clock cycle (next address generator is called sequencer)

    - In-line Sequencing
    - Branch
    - Conditional Branch
    - Subroutine
    - Loop
    - Instruction OP-code mapping

- The general configuration of a microprogrammed control unit is demonstrated in the block diagram

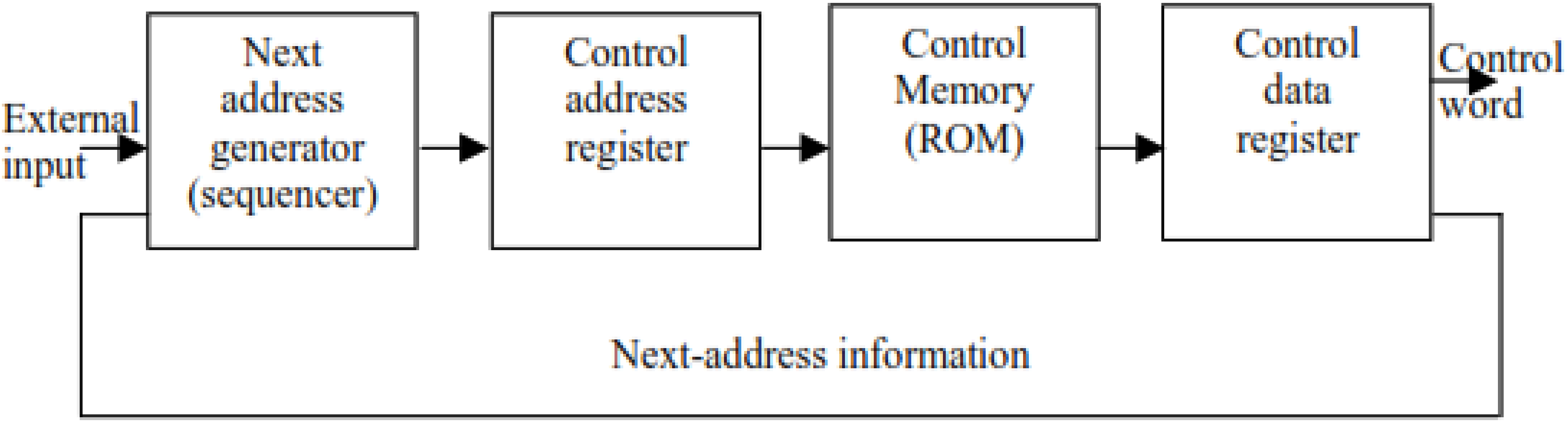


Figure : Microprogrammed control organization.

- The general configuration of a micro-programmed control unit is demonstrated in the block diagram of Figure.
- The control memory is assumed to be a ROM, within which all control information is permanently stored.
- The microinstruction contains a control word that specifies one or more microoperations for the data processor. Once these operations are executed, the control must determine the next address.
- The location of the next microinstruction may be the one next in sequence, or it may be located somewhere else in the control memory.
- While the microoperations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.

- Thus a microinstruction contains bits for initiating microoperations in the data processor part and bits that determine the address sequence for the control memory.
- The next address generator is sometimes called a **micro-program sequencer,** as it determines the address sequence that is read from control memory.
- Typical functions of a micro-program sequencer are incrementing the control address register by one, loading into the control address register an address from control memory, transferring an external address, or loading an initial address to start the control operations.
- The control data register holds the present microinstruction while the next address is computed and read from memory.

Address sequencing

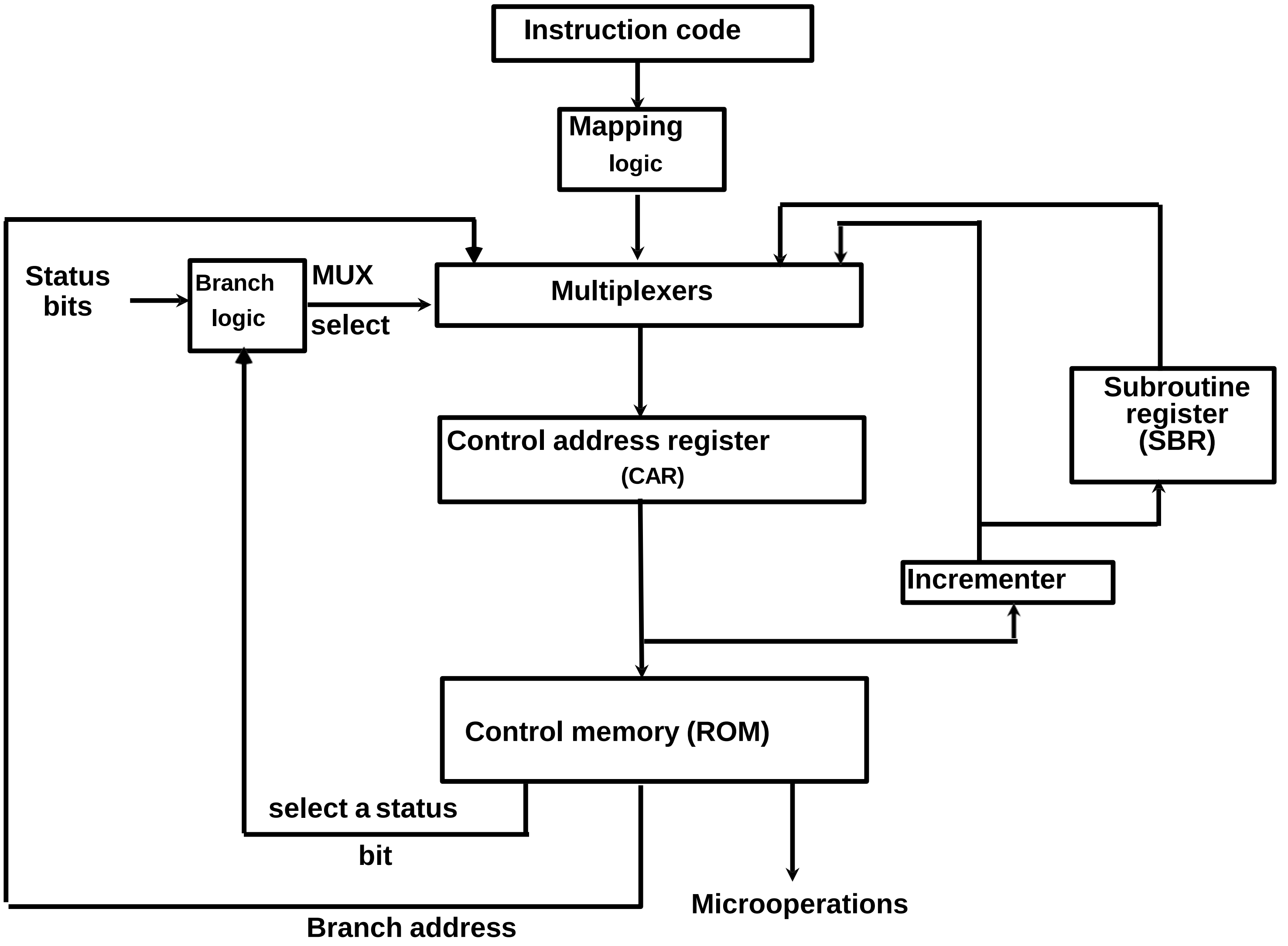


Figure : Selection of Address for controlmemory.

Address Sequencing

- Microinstructions are stored in control memory in groups, with each group specifying a **routine.**
- To appreciate the address sequencing in a micro-program control unit, let us specify the steps that the control must undergo during the execution of a single computer instruction.

Step-1:

- An initial address is loaded into the control address register when power is turned on in the computer.
- This address is usually the address of the first microinstruction that activates the instruction fetch routine.
- The fetch routine may be sequenced by incrementing the control address register through the rest of its microinstructions.
- At the end of the fetch routine, the instruction is in the instruction register of the computer.

Step-2:

- The control memory next must go through the routine that determines the effective address of the operand.
- A machine instruction may have bits that specify various addressing modes, such as indirect address and index registers.
- The effective address computation routine in control memory can be reached through a branch microinstruction, which is conditioned on the status of the mode bits of the instruction.
- When the effective address computation routine is completed, the address of the operand is available in the memory address register.

Step-3:

- The next step is to generate the microoperations that execute the instruction fetched from memory.
- The microoperation steps to be generated in processor registers depend on the operation code part of the instruction.

- Each instruction has its own micro-program routine stored in a given location of control memory.
- The transformation from the instruction code bits to an address in control memory where the routine is located is referred to as a **mapping process.**
- A mapping procedure is a rule that transforms the instruction code into a control memory address.

Step-4:

- Once the required routine is reached, the microinstructions that execute the instruction may be sequenced by incrementing the control address register.
- Micro-programs that employ subroutines will require an external register for storing the return address.
- Return addresses cannot be stored in ROM because the unit has no writing capability.

- When the execution of the instruction is completed, control must return to the fetch routine.
- This is accomplished by executing an unconditional branch microinstruction to the first address of the fetch routine.

In summary, the address sequencing capabilities required in a control memory are:

1. Incrementing of the control address register.
2. Unconditional branch or conditional branch, depending on status bit conditions.
3. A mapping process from the bits of the instruction to an address for control memory.
4. A facility for subroutine call and return.

Steps in Selection of Address

Conditional Branch

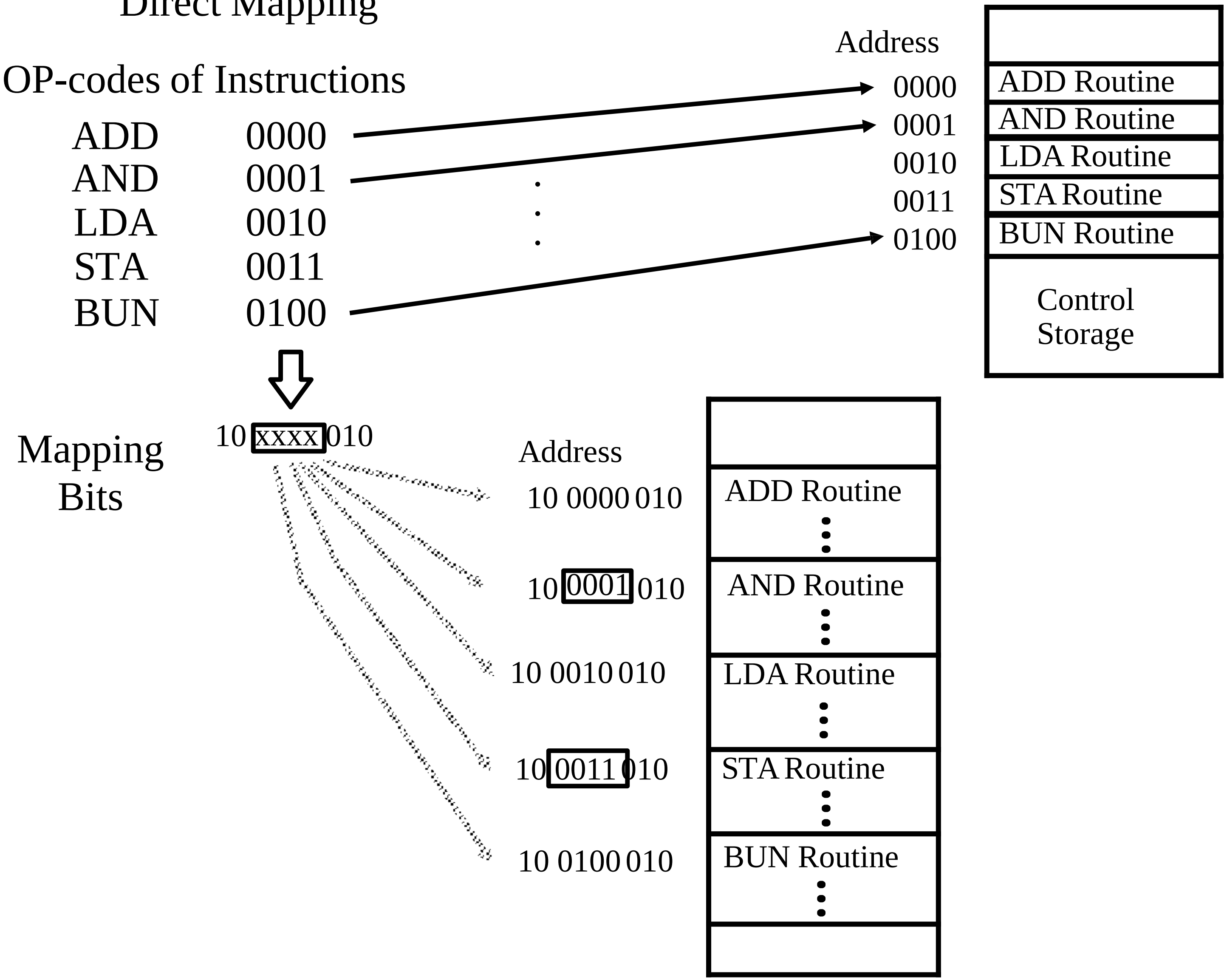
- Branching from one routine to another depending on status bit conditions.
- If Condition is true, then Branch (address from the next address field of the current microinstruction) else Fall Through
- Conditions to Test: O(overflow), N(negative),Z(zero), C(carry), etc.

Unconditional Branch

- Fixing the value of one status bit at the input of the multiplexer to 1

Mapping of instructions

Direct Mapping



OP-codes of Instructions

ADD	0000
AND	0001
LDA	0010
STA	0011
BUN	0100

Address	
0000	ADD Routine
0001	AND Routine
0010	LDA Routine
0011	STA Routine
0100	BUN Routine
	Control Storage

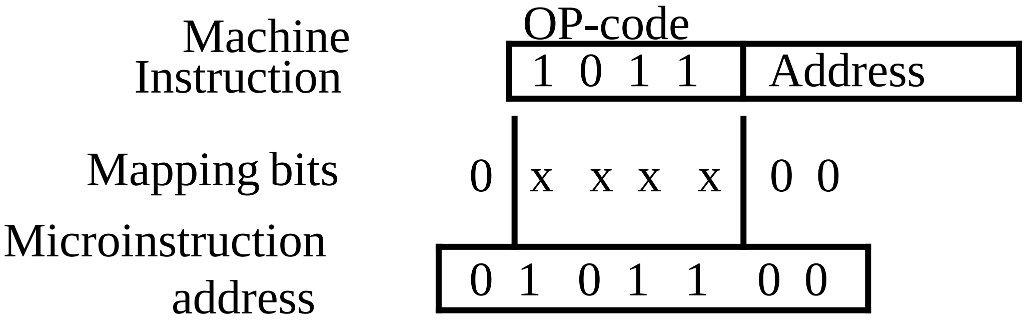
Mapping Bits

10 xxxx 010

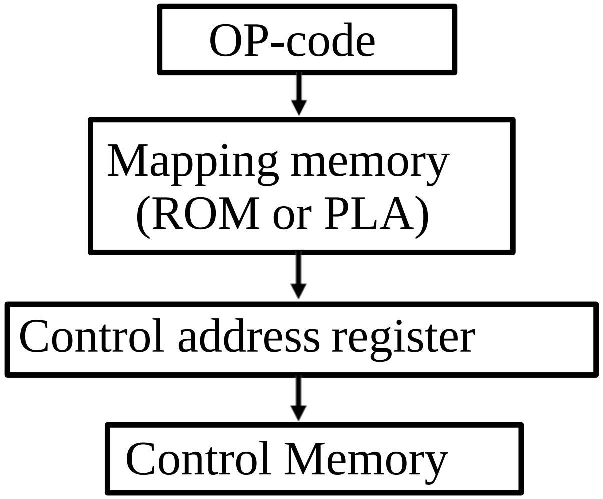
Address	
10 0000 010	ADD Routine
10 0001 010	AND Routine
10 0010 010	LDA Routine
10 0011 010	STA Routine
10 0100 010	BUN Routine

Mapping of instructions to micro-routines

- Mapping from the OP-code of an instruction to the address of the Microinstruction which is the starting microinstruction of its execution microprogram



Mapping function implemented by ROM or PLA



Subroutines

- Subroutines are programs that are used by other routines to accomplish a particular task. A subroutine can be called from any point within the main body of the microprogram.
- For example, the sequence of microoperations needed to generate the effective address of the operand for an instruction is common to all memory reference instructions. This sequence could be a subroutine that is called from within many other routines to execute the effective address computation.
- The subroutine register can then become the source for transferring the address for the return to the main routine.
- The best way to structure a register file that stores addresses for subroutines is to organize the registers in a last-in, first-out (LIFO) stack

Microprogram Example

- Once the configuration of a computer and its microprogrammed control unit is established, the designer's task is to generate the microcode for the control memory. This code generation is called microprogramming and is a process similar to conventional machine language programming.
- To appreciate this process, we present here a simple digital computer and show how it is microprogrammed

Computer Configuration

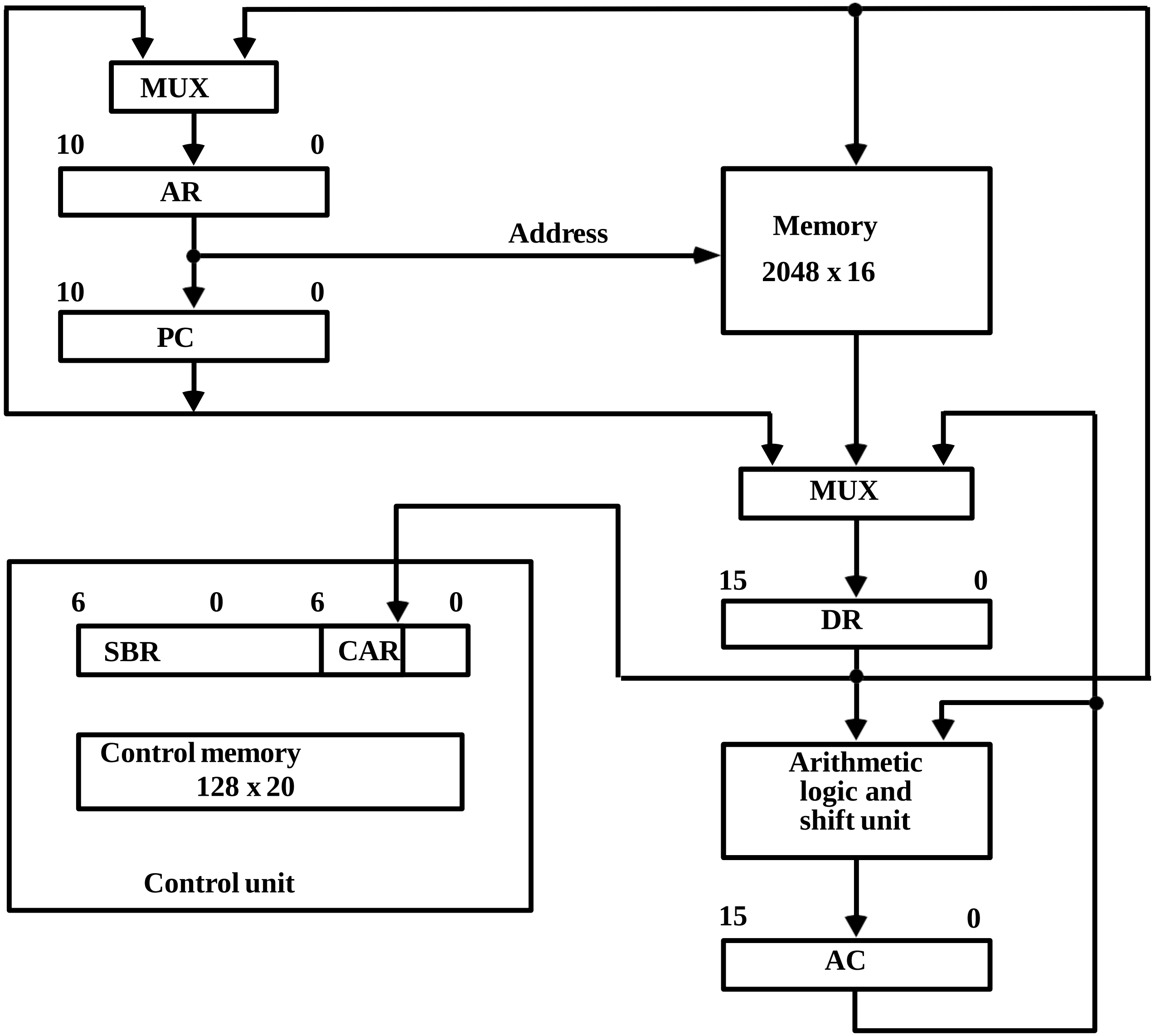


Figure: Computer hardware configuration.

- Consists of two memory units: a main memory for storing instructions and data, and a control memory for storing he microprogram.
- Four register are associated with the processor unit and two with the control unit.
- The processor registers are program counter PC, address register AR, data register DR, and accumulator register DR, and accumulator register AC. The function of these registers is similar to the basic computer.
- The control unit has a control address register CAR and a subroutine register SBR.
- The control memory and its registers are organized as a micro programmed control unit.
- The transfer of information among the register in the processor is done through multiplexers rather than a common bus.

- DR can receive information from AC, PC, or memory.
- AR can receive information from PC or DR, PC can receive information only from AR.
- The arithmetic, logic, and shift unit performs microoperations with data from AC and DR and places the result inAC.
- Note that memory receive its address from AR. Input data written to memory come from DR, and data read from memory can go only to DR.

Microinstruction Format

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

F1, F2, F3: Microoperation fields

CD: Condition for branching

BR: Branch field

AD: Address field

Figure : Micro-instructions code format (20 bits).

- The microinstruction format for the control memory is shown in figure . The 20 bits of the microinstruction are divided into four functional parts as follows:
  - The three fields F1, F2, and F3 specify microoperations for the computer. The microoperations are subdivided into three fields of three bits each. The three bits in each field are encoded to specify seven distinct microoperations. This gives a total of 21 microoperations.
  - The CD field selects status bit conditions.
  - The BR field specifies the type of branch to be used.
  - The AD field contains a branch address. The address field is seven bits wide, since the control memory has 128 = 2^7 words.

Microinstruction field descriptions – F1,F2,F3

F1	Microoperation	Symbol
000	None	NOP
001	AC ← AC + DR	ADD
010	AC ← 0	CLRAC
011	AC ← AC + 1	INCAC
100	AC ← DR	DRTAC
101	AR ← DR(0-10)	DRTAR
110	AR ← PC	PCTAR
111	M[AR] ← DR	WRITE

F2	Microoperation	Symbol
000	None	NOP
001	AC ← AC - DR	SUB
010	AC ← AC ∨ DR	OR
011	AC ← AC ∧ DR	AND
100	DR ← M[AR]	READ
101	DR ← AC	ACTDR
110	DR ← DR + 1	INCDR
111	DR(0-10) ← PC	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	AC ← AC ⊕ DR	XOR
010	AC ← AC’	COM
011	AC ← shl AC	SHL
100	AC ← shr AC	SHR
101	PC ← PC + 1	INCPC
110	PC ← AR	ARTPC
111	Reserved	

Microinstruction field descriptions – CD,BR

CD	Condition	Symbol	Comments
00	Always = 1	U	Unconditional branch
01	DR(15)	I	Indirect address bit
10	AC(15)	S	Sign bit of AC
11	AC = 0	Z	Zero value in AC

BR	Symbol	Function

00	JMP	CAR ← AD if condition = 1
		CAR ← CAR + 1 if condition = 0

01	CALL	CAR ← AD, SBR ← CAR + 1 if condition = 1
		CAR ← CAR + 1 if condition = 0

10	RET	CAR ← SBR (Return from subroutine)
11	MAP	CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

Symbolic Microinstructions

- Symbols are used in microinstructions as in assembly language.
- A symbolic microprogram can be translated into its binary equivalent by a microprogram assembler.

Sample Format

Five fields: label ; micro-ops ; CD ; BR ; AD

Label: may be empty or it may specify a symbolic address terminated with a colon (:).

Micro-ops: consists of one, two, or three symbols separated by commas

CD: one of {U, I, S, Z}, where U: Unconditional Branch

I: Indirect address bit

S: Sign of AC

Z: Zero value in AC

BR: contains one of {JMP, CALL, RET, MAP}

AD: specifies a value for the address field of the microinstructions in one of three possible ways :

- With a symbolic address, which must also appear as a label.
- With the symbol NEXT to designate the next address in sequence.
- When the BR field contains a RET or MAP symbol, the AD field is left empty and is converted to seven zeros by the assembler.

Symbolic microprogram - fetch routine

- During FETCH, Read an instruction from memory and decode the instruction and update PC.
- Sequence of microoperations in the fetch cycle:

AR ← PC

DR ← M[AR], PC ← PC + 1

AR ← DR(0-10), CAR(2-5) ← DR(11-14), CAR(0,1,6) ← 0

- Symbolic microprogram for the fetch cycle:

```
            ORG 64
FETCH:      PCTAR          U   JMP  NEXT
            READ, INCPC    U   JMP  NEXT
            DRTAR          U   MAP
```

- The translation of the symbolic microprogram to binary produces the following binary microprogram.

Binary						
address	**F1**	**F2**	**F3**	**CD**	**BR**	**AD**
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

- Control Storage: 128 20-bit words
- The first 64 words: Routines for the 16 machine instructions
- The last 64 words: Used for other purpose (e.g., fetch routine and other subroutines)
- Mapping: OP-code XXXX into 0XXXX00, the first address for the 16 routines are 0(0 0000 00), 4(0 0001 00), 8, 12, 16, 20, ..., 60

Partial Symbolic Microprogram

Label	Microops	CD	BR	AD
	ORG 0			
ADD:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
	ORG 4			
BRANCH:	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
	ORG 8			
STORE:	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 12			
EXCHANGE:	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 64			
FETCH:	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
	DRTAR	U	MAP	
INDRCT:	READ	U	JMP	NEXT
	DRTAR	U	RET	

The execution of the ADD instruction is carried out by the microinstructions at addresses 1 and 2. The first microinstruction reads the operand from memory into DR . The second microinstruction performs an add microoperation with the content of DR and AC and then jumps back to the beginning of

the fetch routine.

Binary Microprogram

- The symbolic microprogram is a convenient form for writing microprograms in a way that people can read and understand. But this is not the way that the microprogram is stored in memory.
- The symbolic microprogram must be translated to binary either by means of an assembler program or by the user if the microprogram is simple enough.

Binary microprogram

	Address		Binary Microinstruction					
Micro Routine	Decimal	Binary	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	00	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

Design of control unit

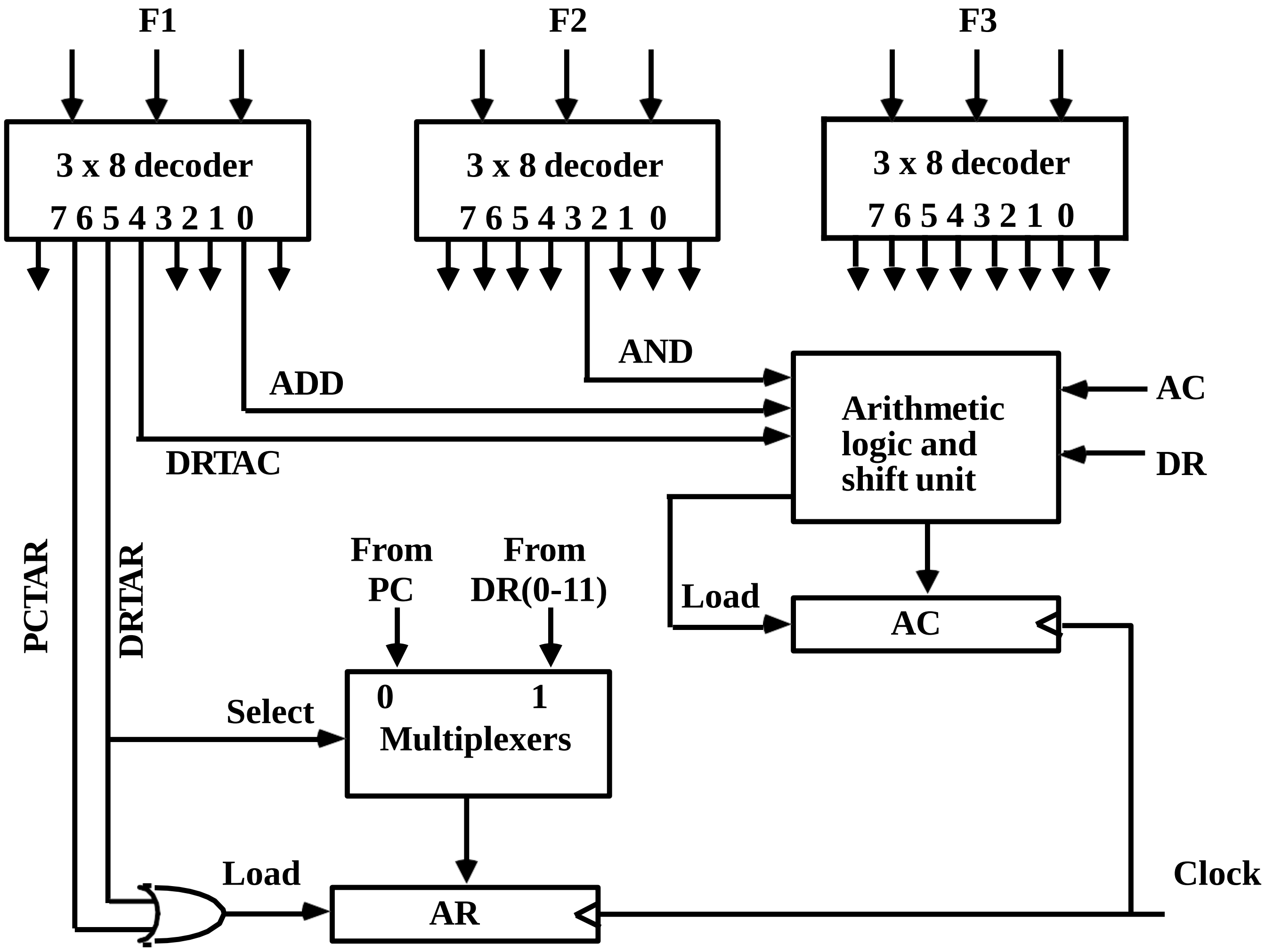


Figure : Decoding of Microoperation Fields.

- The bits of the microinstruction are usually divided into fields, with each field defining a distinct, separate function.
- Figure shows the three decoders and some of the connections that must be made from their outputs. Each of the three fields of the microinstruction presently available in the output of control memory are decoded with a 3 × 8 decoder to provide eight outputs.
- From F1 =7 outputs, F2=7 outputs and F3=6 outputs total 20 outputs from decoder.
- For example, when F1 = 101 (binary 5), the next clock pulse transition transfers the content of DR (010) to AR (symbolized by DRTAR).
- Similarly, when F1 = 110 (binary 6) there is a transfer from PC to AR (symbolized by PCTAR).
- outputs 5 and 6 of decoder F1 are connected to the load input of AR so that when either one of these outputs is active, information from the multiplexers is transferred to AR.

- The multiplexers select the information from DR when Output 5 is active and from PC when output 5 in inactive. The transfer into AR occurs with a Clock pulse transition only when output 5 or output 6 of the decoder are active.
- The other outputs of the decoders that initiate transfers between registers must be connected in a similar fashion.
- The arithmetic logic shift unit can be designed Instead of using gates to generate the control signals marked by the symbols AND, ADD, and DR, , these inputs will now come from the outputs of the decoders associated with the symbols AND, ADD, and DRTAC.

Microprogram sequencer

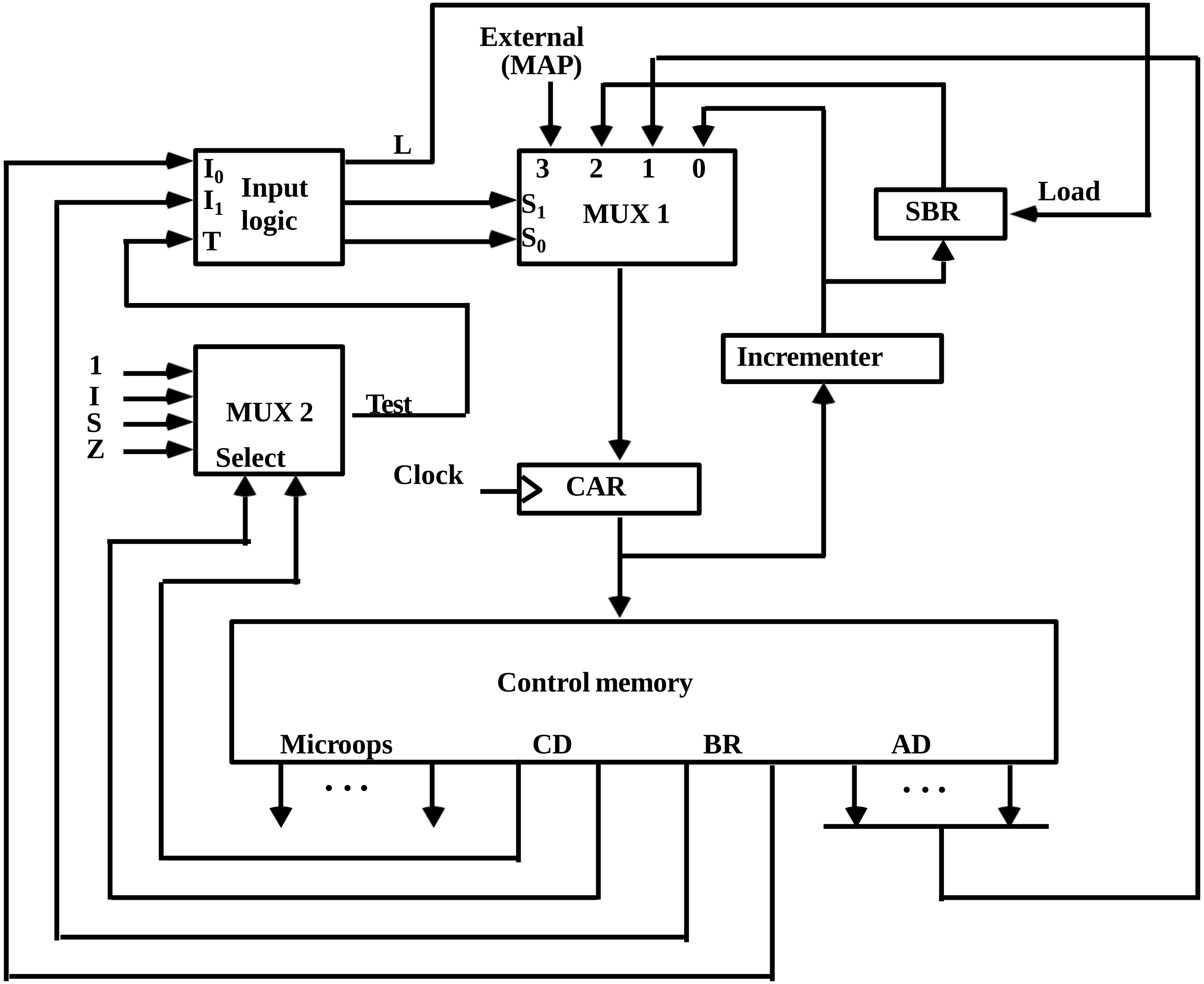


Figure : Microprogram sequencer for a control memory.

- The submit of the micro programmed control unit which presents an address to the control memory is called microprogram sequencer.
- The input logic circuit has three inputs, I_0, I_1, and T, and three outputs, S_0, S_1 and L. variables S_0 and S_1 select one of the source addresses for CAR.
- Variable L enables the load input in SBR. The binary values of the two selection variables determine the path in the multiplexer.
- Four inputs to the MUX 1 ,select one address and store into CAR. MUX 2 test the value of selected status bit and result to input logic circuit.
- For example, with S_1 S_0 = 10, multiplexer input number 2 is selected and establishes a transfer path from SBR to CAR.
- Note that each of the four inputs as well as the output of MUX 1 contains a 7-bit address.
- The truth table for the input logic circuit is shown in the table. inputs I_1 and I_0 are identical to the bit values in the BR field.

- The bit values for S_1 and S_0 are determined from the stated function and the path in the multiplexer that establishes the required transfer.
- The subroutine register is loaded with the incremented value of CAR during a call microinstruction (BR = 01) provided that the status bit condition is satisfied (T = 1). The truth table can be used to obtain the simplified Boolean functions for the input logic:

$$S_1 = I_1$$

$$S_0 = I_1 I_0 + I_1' T$$

$$L = I_1' I_0 T$$

BR	Input	MUX 1	Load SBR
Field	I_1I_0T	S_1S_0	L
0 0	000	00	0
0 0	001	01	0
0 1	010	00	0
0 1	011	01	1
1 0	10x	10	0
1 1	11x	11	0

Table : Input logic truth table for microprogram sequencer.