Micro-programmed control

- CONTROL MEMORY
- ADDRESS SEQUENCING
- MICRO-PROGRAM EXAMPLE
  - Computer Configuration
  - Microinstruction Format

Control memory

- Control Memory (Control Storage: CS)
  - Storage in the micro-programmed control unit to store the micro-program.
- Control word
  - It is a string of control variables (0's and 1's) occupying a word in control memory.
- Micro-program
  - Program stored in control 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 – contains all the control information required for one clock cycle
  - Sequencing Word - Contains information needed to decide the next microinstruction address

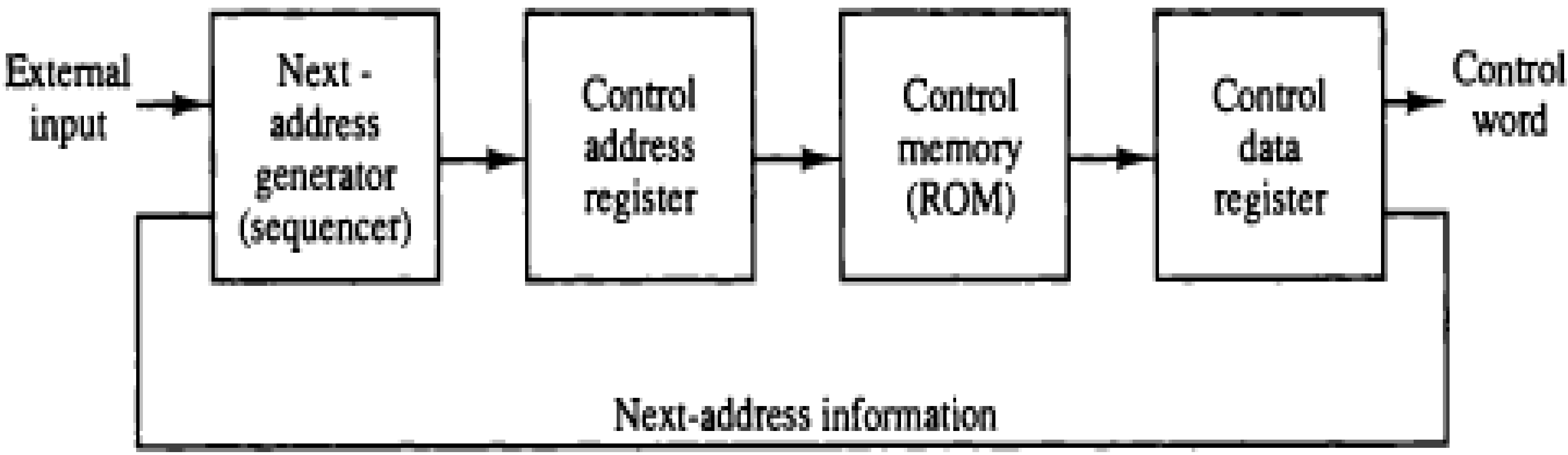
Control memory (contd..)

- Writable Control Memory (Writable Control Storage: WCS)
  - CS whose contents can be modified:
  - Micro-program can be changed
  - Instruction set can be changed or modified
- A computer that employs a micro-programmed control unit will have two separate memories**: main memory and a control memory.**
- The user's program in main memory consists of machine instructions and data whereas control memory holds a fixed micro program that cannot be altered by the user.
- Each machine instruction initiates a series of microinstructions in control memory.

Control memory (contd..)

- The general configuration of a micro-programmed control unit is demonstrated in the following block diagram:

Figure Microprogrammed control organization.



Control memory (contd..)

- Dynamic Microprogramming
  - Computer system whose control unit is implemented with a micro-program in WCS.
  - Micro-program can be changed by a systems programmer or a user
- Control Address Register
  - Control address register contains address of microinstruction.
- Control Data Register
  - Control data register contains microinstruction.
- Sequencer
  - The device or program that generates address of next microinstruction to be executed is called sequencer.

Address Sequencing

- Each computer instruction has its own micro-program routine in control memory to generate the micro-operations that execute the instruction.
- Process of finding address of next micro-instruction to be executed is called address sequencing.
- Address sequencer must have capabilities of finding address of next micro-instruction in following situations:
  - In-line Sequencing
  - Unconditional Branch
  - Conditional Branch
  - Subroutine call and return
  - Looping
  - Mapping from instruction op-code to address in control memory.

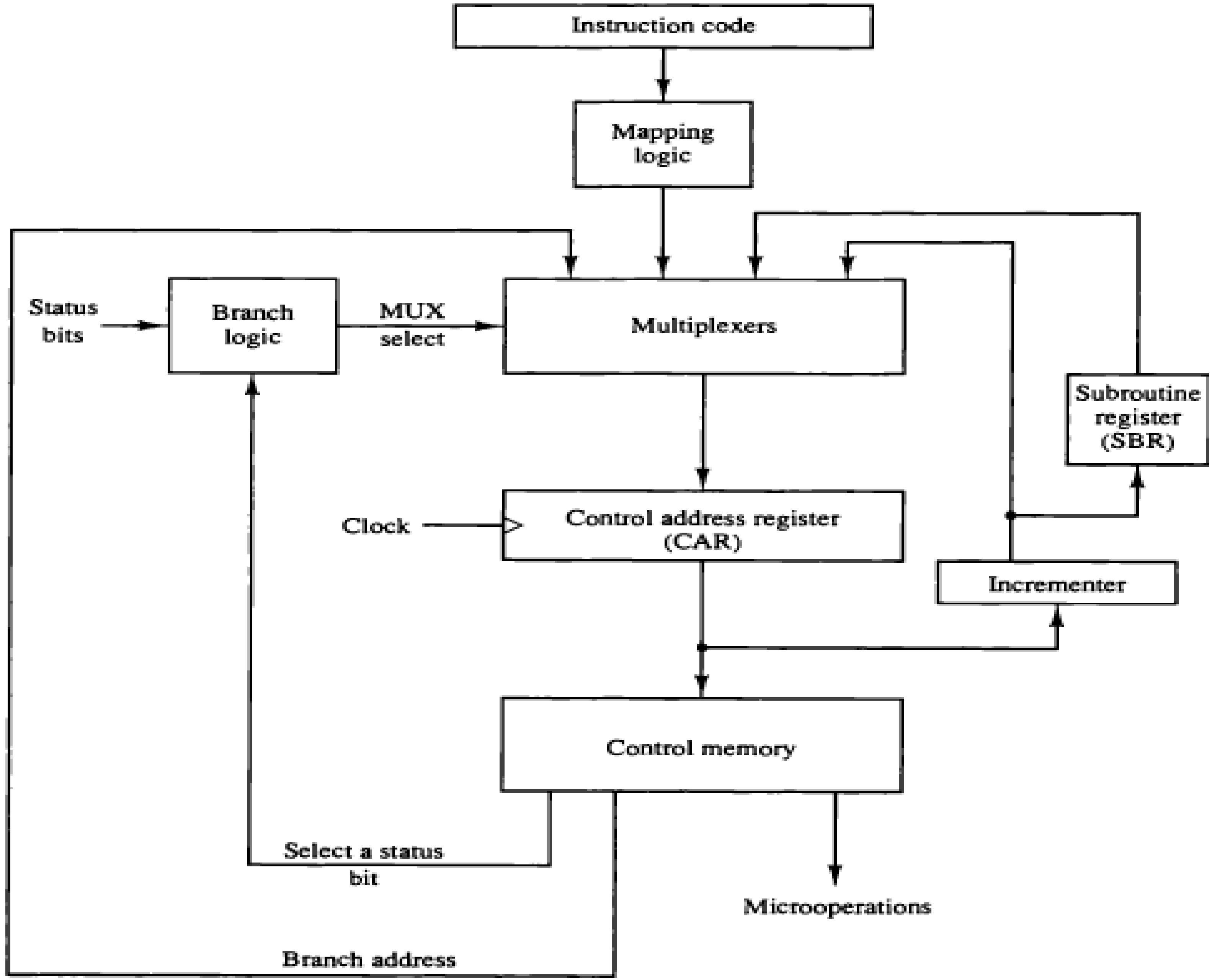


Figure Selection of address for control memory.

Address Sequencing (contd..)

- Control address register receives address of next micro instruction from different sources.
- Incrementer simply increments the address by one
- In case of branching branch address is specified in one of the field of microinstruction.
- In case of subroutine call return address is stored in the register SBR which is used when returning from called subroutine.

Conditional jumping

- Simplest way of implementing branch logic hardware is to test the specified condition and branch to the indicated address if condition is met otherwise address resister is simply incremented .
- If Condition is true, h/w set the appropriate field of status register to 1.
- Conditions are tested for O (overflow), N (negative), Z (zero), C (carry), etc.

Unconditional Jumping

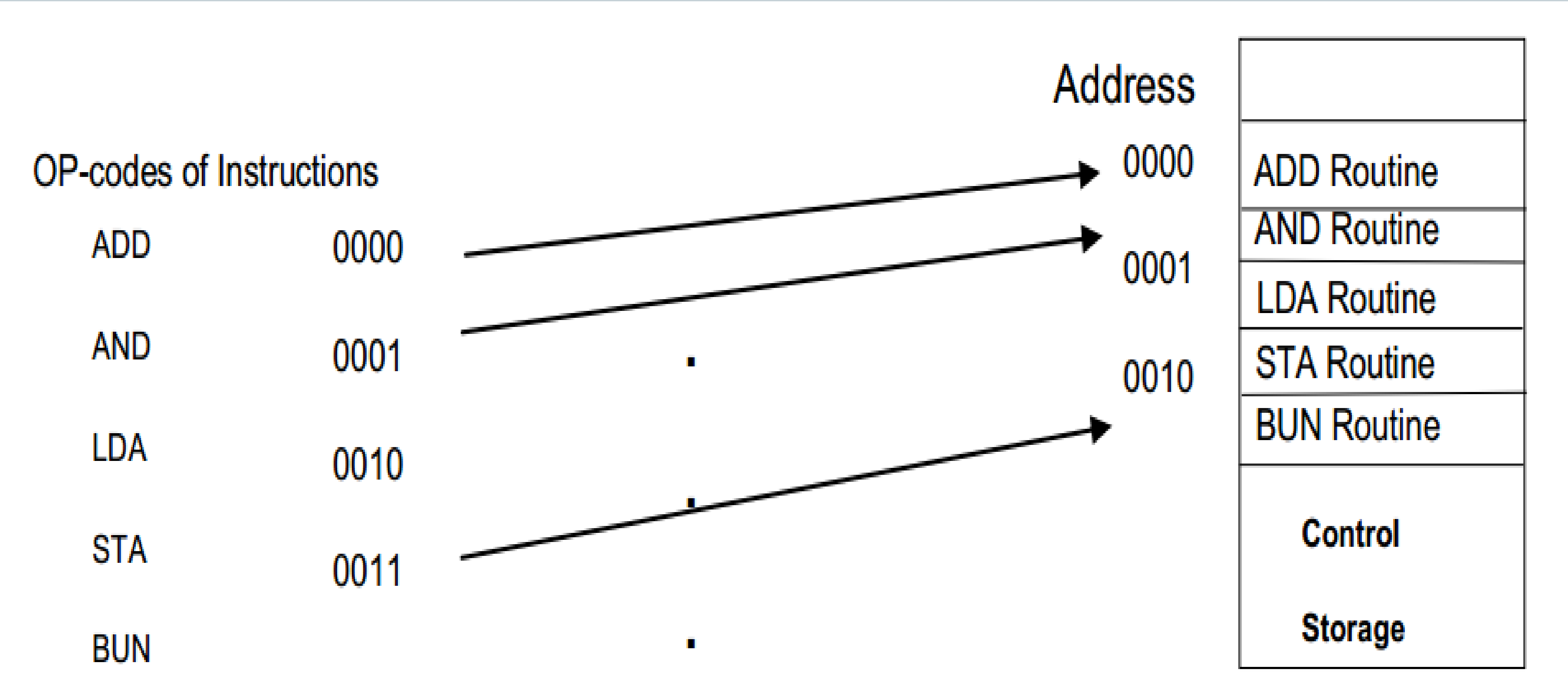
- Fix the value of one status bit at the input of the multiplexer to 1.
- So that, branching can always be done.

Mapping

- Assuming operation code of 4-bits which can specify 16 (2^4) distinct instructions.
- Assume further and control memory has 128 words, requiring an address of 7-bits. Now we have to map 4-bit operation code into 7-bit control memory address.
- Thus, we have to map Op-code of an instruction to the address of the Microinstruction which is the starting microinstruction of its subroutine in memory.

Direct mapping

- Directly use opcode as address of Control memory



Another approach of direct mapping:

- Transfer Op-code bits to use it as an address of control memory.

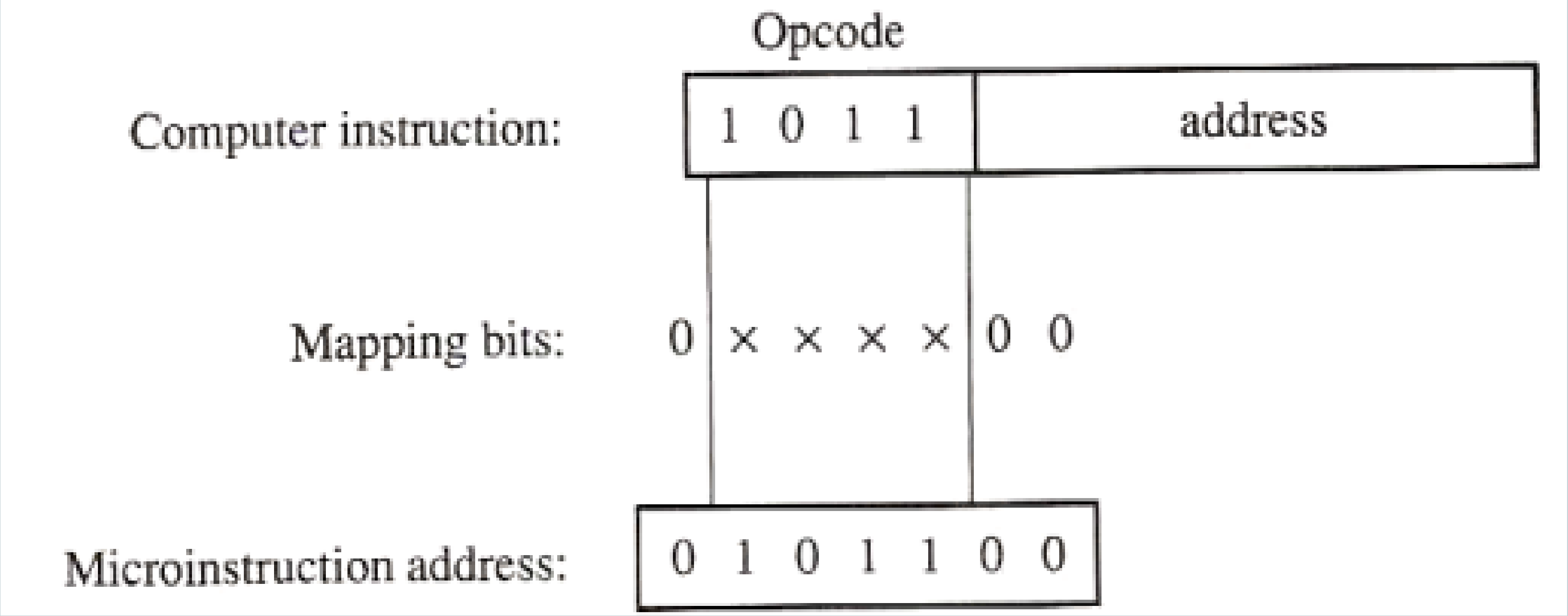
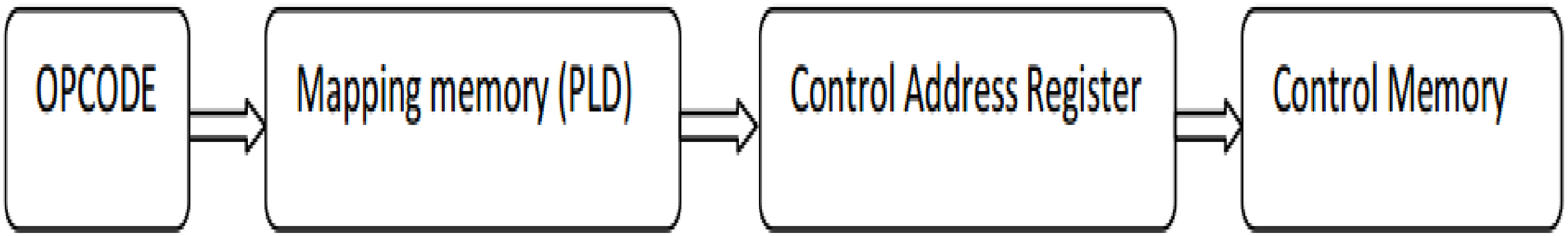


Fig: mapping from instruction code to microinstruction address

Extended idea: Mapping function implemented by ROM or PLD(Programmable Logic Device)

- Use op-code as address of ROM where address of control memory is stored and than use that address as an address of control memory.
- This provides flexibility to add instructions for control memory as the need arises.



Subroutines

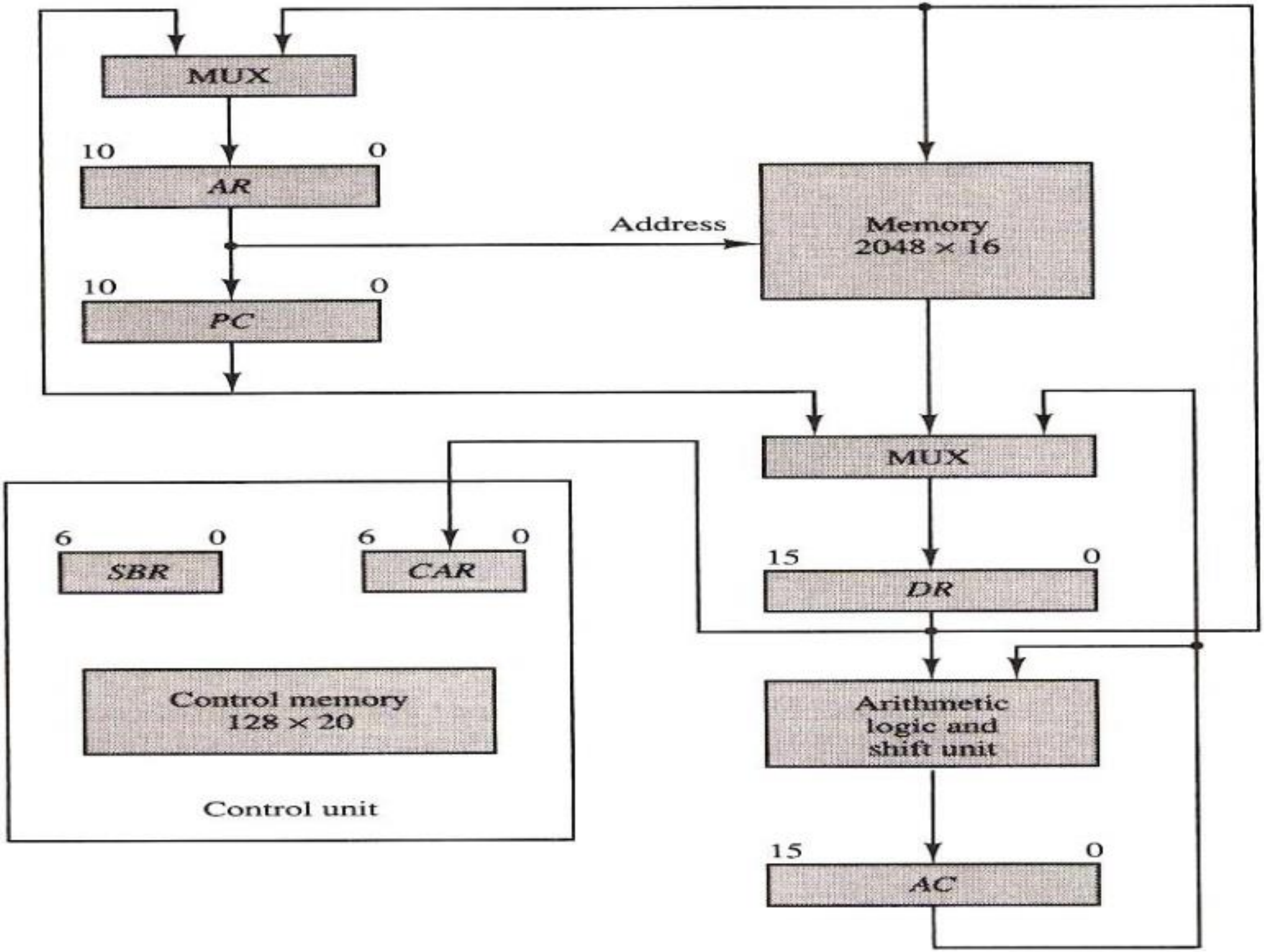
- Subroutines are programs that are used by another program to accomplish a particular task.
- Microinstructions can be saved by employing subroutines that use common sections of micro code.
- Example: the sequence of micro-operations needed to generate the effective address is common to all memory reference instructions.
- Thus, this sequence could be a subroutine that is called from within many other routines to execute the effective address computation.
- Subroutine resister is used to save a return address during a subroutine call which is organized in LIFO (last in, first out) stack

Micro-program (An example)

- Once we have a configuration of a computer and its micro-programmed control unit, the designer generates the microcode for the control memory.
- Code generation of this type is called micro-programming and is similar to conventional machine language programming.
- We assume here a simple digital computer similar (but not identical) to Manos' basic computer.

Computer configuration

- Block diagram is shown below; it consists of two memory units: a main memory for storing instructions and data, and a control memory for storing the micro-program.
- 4 resisters are with processor unit and 2 resisters with the control unit.


MUX
10
0
AR
Address
Memory
2048 × 16
10
0
PC
MUX
6
0
SBR
6
0
CAR
15
0
DR
Control memory
128 × 20
Arithmetic
logic and
shift unit
Control unit
15
0
AC

Microinstruction Format

- Micro-instruction in control memory has 20-bit format divided into 4 functional parts as shown below.

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

Microinstruction Format

- Each micro-operation below is defined using resister transfer statements and is assigned a symbol for use in symbolic micro-program.

Description of CD

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

Description of BR

BR	Symbol	Function
00	JMP	$CAR \leftarrow AD$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
01	CALL	$CAR \leftarrow AD$, $SBR \leftarrow CAR + 1$ if condition = 1
		$CAR \leftarrow CAR + 1$ if condition = 0
10	RET	$CAR \leftarrow SBR$ (Return from subroutine)
11	MAP	$CAR(2\text{–}5) \leftarrow DR(11\text{–}14)$, $CAR(0,1,6) \leftarrow 0$

- CD (condition) field consists of two bits representing 4 status bits and BR (branch) field (2-bits) used together with address field AD, to choose the address of the next microinstruction.

Microinstruction fields (F1, F2, F3)

F1	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0\text{–}10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0\text{-}10) \leftarrow PC$	PCTDR

F3	Microoperation	Symbol
000	None	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow \text{shl } AC$	SHL
100	$AC \leftarrow \text{shr } AC$	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

Microinstruction fields (F1, F2, F3)

- Micro-operations are subdivided into three fields of 3-bits each.
- These 3 bits are used to encode 7 different micro-operations.
- No more than 3 micro-operations can be chosen for a microinstruction, one for each field.
- If fewer than 3 micro-operations are used, one or more fields will contain 000 for no operation.