# Address sequencing

→ Microinstructions are stored in control memory in groups, with each group specifying a routine. Each computer instruction has it's own microprogram routine to generate these microoperations. The hardware that controls the address. Sequencing of the control memory must be capable of sequencing the microinstructions within a routine & be able to branch from one to another.

Steps the control must undergo during the execution of a single computer instruction:

① Load ~~a~~ an initial address into the CAR when power is turned on in the computer. This address is usually the address of the first microinstructions that activate the instruction fetch routine. i.e, IR holds instruction.

② The control memory then goes through the routine to determine the effective address of the operand - AR holds operand address.

③ The next step is to generate the microoperations that executes the instruction by considering the upcode & applying a mapping. ~~The next~~

④ After execution, control must return to the fetch routine ~~at~~ by executing an unconditional branch.

The microinstruction in control memory contains a set of bits to initiate microoperations in computer registers & other bits to specify the method by which the next address is obtained.

Scanned with CamScanner

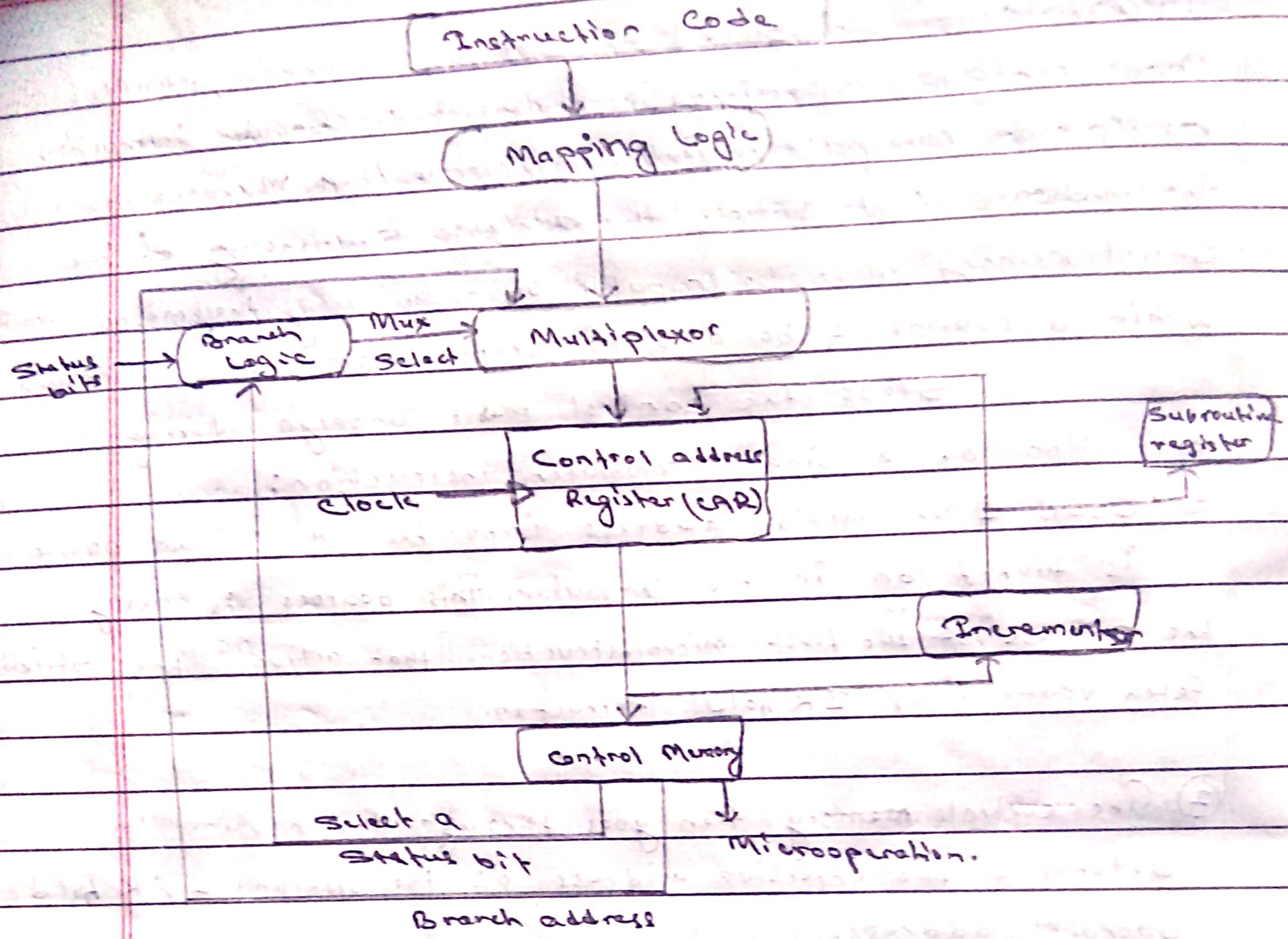


fig: Selection of address for control memory.

① conditional Branching.

Conditional Branching is obtained by using parts of the microinstruction to select a specific status bit in order to determine its condition. The status conditions are special bits in the system that provide parametre information such as the carry out of an Adder, The sign bit of a number, the MOD bit of an instruction, on I/O status condition. The status bits together with the field in the micro instruction that specify a branch address, control the branch logic. The Branch logic, tests the condition, it ~~[illegible]~~ ~~[illegible]~~ met then

Branches, otherwise, increment the CAR. If there are 8 status ~~bit~~ bit conditions then 3 bits in the microinstruction are used to specify the condition & provide the selection variables for the multiplexor. For unconditional Branching fixed the value of 1 status bit to be load the branch address from control memory ~~to~~ into CAR.

② Mapping of Instruction.

→ A special type of branch exists when a microinstruction Specifies a branch to the first word in control memory Where a microprogram routine is located. The status bit for this type of branch are the bits in the opcode. Assume an opcode of 4 bits & control memory of 128 Locations. So, the mapping process convert the 4 bit opcode to a 7 bit address for control memory. This provides for each computer instruction A microprogram routine with a capacity of 4 microinstructions

computer instruction: | 0 1 1 1 | address |

mapping bits → 0 | x x x x | 00

Microinstruction address → | 0 1 0 1 1 0 0 |

fig: Mapping from instruction code to microinstruction address.

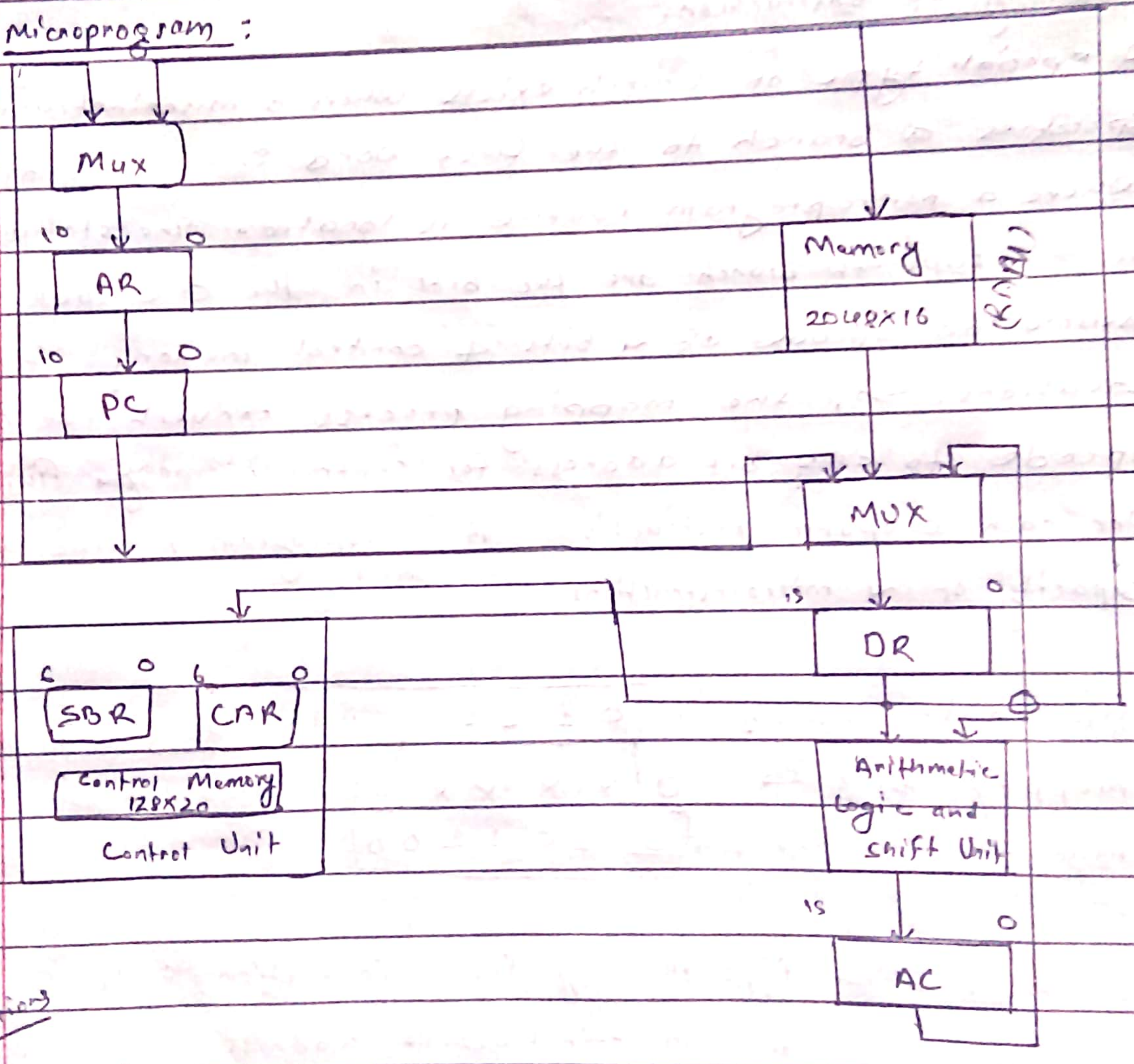
③ Subroutine:

Subroutines are programmes that are used by other routines to accomplish a particular task & can be called from any point within the main body of the microprogram. frequently many microprograms contain identical section of code. Microinstructions can be saved by imploying subroutines.

Scanned with CamScanner

That use common sections of microcode. Microprogram that use subroutine must have a provisions for storing the ~~[illegible]~~ return address during a subroutine call & restoring the address during a subroutine return. A subroutine register is used as the source and destination for the address.

# Microprogram :



2048 - locations

It^contains^ 2 memory unit (A main memory for storing instructions & data) & a control memory for storing microprogram. 4 registers (PC, AR, DR, AC) with the processor unit & 2 registers (SBR & CAR) with the control unit. The control memory & its register are organized as a microprogram control unit. The transfer of information among the register in the processor is done through multiplexers rather than a common Bus.

DR recieves information from AC, PC or memory. AR recieves information from DR & PC. PC recieves information from AR. The arithmetic, logic & shift units performs microoperations with the data from AC & DR & place the result ~~[illegible]~~ in AC. Memory recieves it's address from AR. ~~Memory recieves its address from~~. Input data retain to memory comes from DR. Data read from memory can go only to DR.

# Instruction format:

| 15 | 14 | 10 0 |
|---|---|---|
| I | opcode | Address |

fig (A)

| Symbol | Opcode | Description |
|---|---|---|
| ADD | 0000 | AC ← AC + M[EA] |
| BRANCH | 0001 | if (AC>0) then (PC ← EA) |
| STORE | 0010 | M[EA] ← AC |
| EXCHANGE | 0011 | AC ← M[EA], M[EA] ← AC |

fig(b) Basic computer instruction

1-bit for addressing mode
4 bit for ~~address~~ opcode
11 bit for address

4 bit opcode produce 16 possible memory reference instructions out of them 4 was explained ~~[illegible]~~ as example in fig b.

## # Design of Basic computer.

# A Basic computer consists of the following hardware component:

- A memory unit with 4096 words of 16 bit.
- It has 9 registers:

① AR (address reg.) ② PC (program counter), ③ DR (Data reg.)
④ AC (Accumulator), ⑤ IR (Instruction Reg.), ⑥ TR (Temp. Reg.)
⑦ OUTR (output Reg.), ⑧ INPR (Input Reg.) ⑨ SC (sequence counter)

- flipflops
- A 16-bit common Bus
- control logic gates
- Adder & logic circuit connected to the input of AC.

## # Design of Accumulator logic:

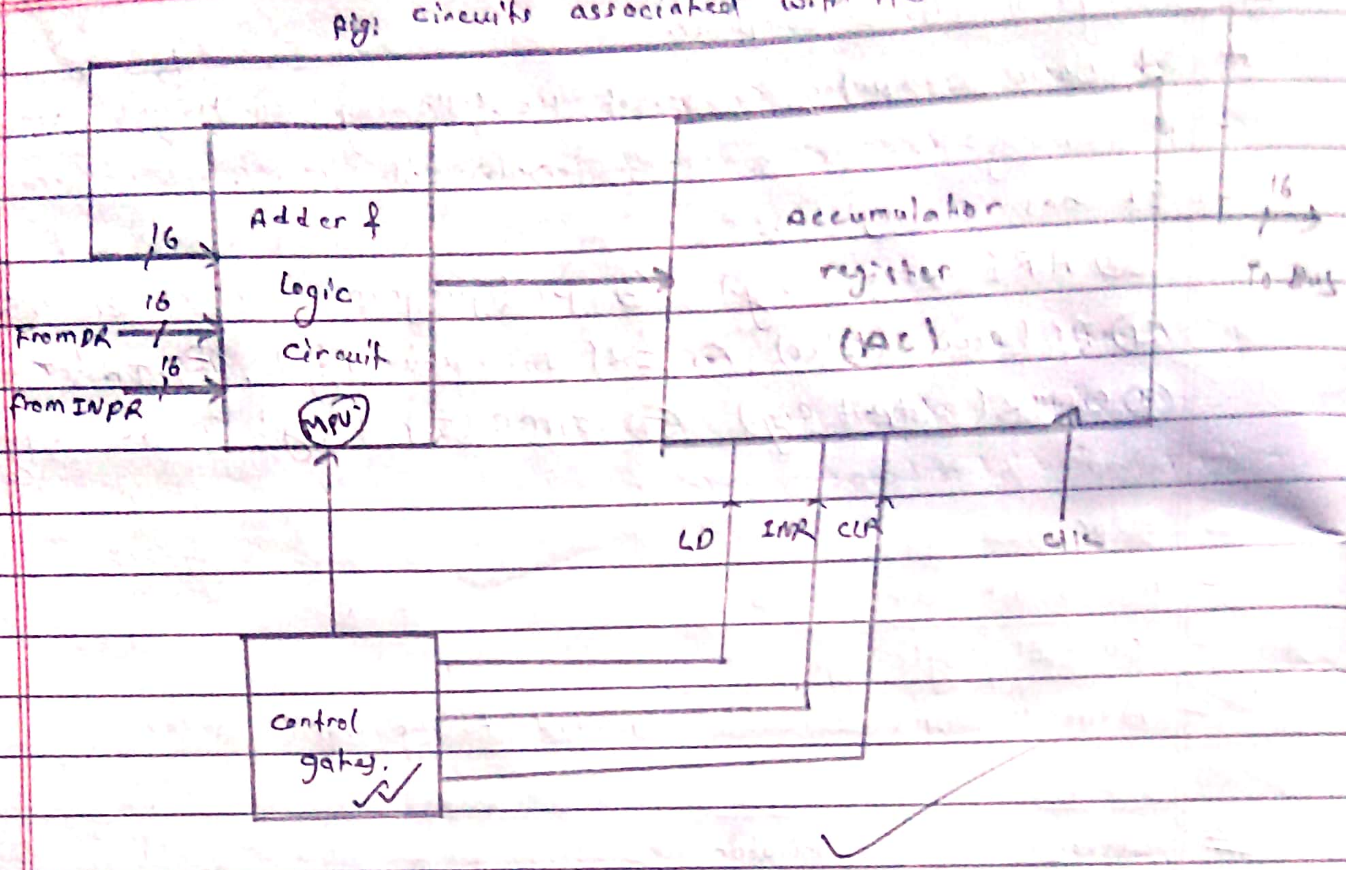
- The circuit Associated with the AC register are shown in fig.
- The adder & logic circuit has 3 set of inputs.

- One set of 16 inputs comes from output of AC.
- Another set of 16 inputs come from the data register (DR).
- The third set of ~~input~~ eight inputs comes from the input register (INPR)
- The outputs of the adder and logic unit provide the data input for register. In addition, it is necessary to include logic gates for controlling the LD, INR & CLR in the register & for controlling the operation of adder & Logic circuit.

Scanned with CamScanner

Date / /

Fig: circuits associated with AC



# Control of AC:

- The Gate structure that controls the LD, INR & CLR inputs of AC is as shown in fig.

- The output of $\overset{AND}{\cancel{the}}$ gate that generates this control function is connected to the CLR of the register.

- similarly, the output of the gate that implements the increment of micro-operation is connected to the INR of the register.

- The other seven microoperations are generated in the adder & logic unit (circuit) and are loaded into at AC at proper time.

- The outputs of the gates for each control function is marked with a symbolic name. These outputs are used in the design of the adder and logic circuit.

Date / /

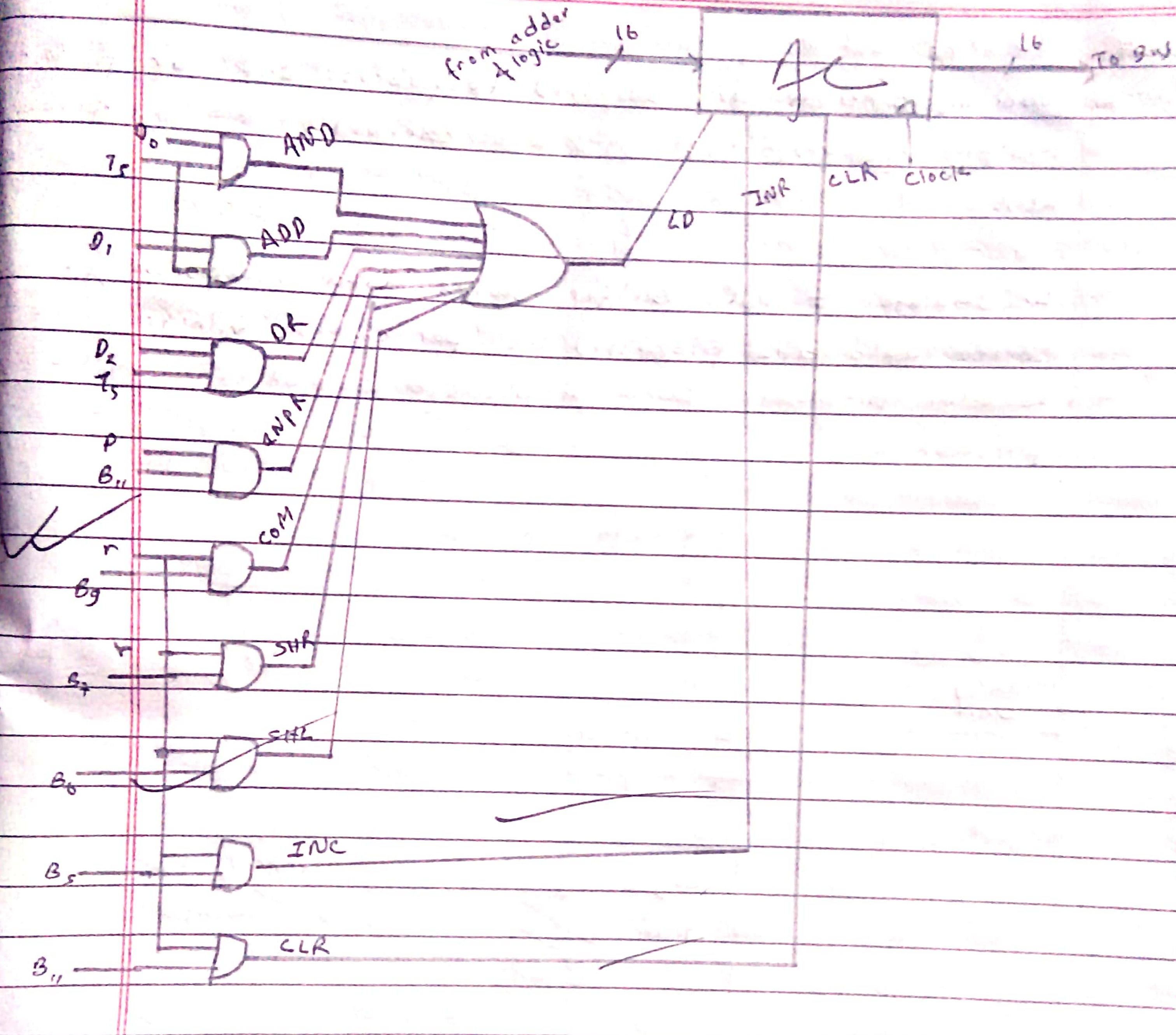


fig: Control of AC, Gate structure.