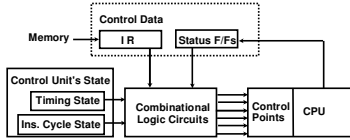
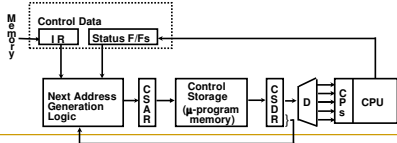


## Microprogrammed Control

### Combinational Logic Circuits (Hard-wired)



### Microprogram



## Terminology

- **Microprogram**
  - Program stored in memory that generates all the control signals required to execute the instruction set correctly
  - Consists of microinstructions
- **Microinstruction**
  - Bits that sequence one or more microoperations
  - Vocabulary to write a microprogram
  - Etc...

## Terminology

- **Control Memory (Control Storage: CS)**
  - Storage in the microprogrammed control unit to store the microprogram
- **Writeable Control Memory (Writeable Control Storage: WCS)**
  - CS whose contents can be modified
    - Allows the microprogram can be changed
    - Instruction set can be changed or modified
- **Dynamic Microprogramming**
  - Computer system whose control unit is implemented with a microprogram in WCS
  - Microprogram can be changed by a systems programmer or a user

## How Does It Work?

- An initial address is loaded into the control address register when the power is turned on
  - This is usually the address of the first microinstruction that activates the instruction fetch
  - At the end of the fetch routine the instruction is in IR
- The control memory next must go through the routine that determines the effective address of the operand
  - Comment on the machine instruction!
  - When the effective address computation routine is completed the address of the operand is available in AR

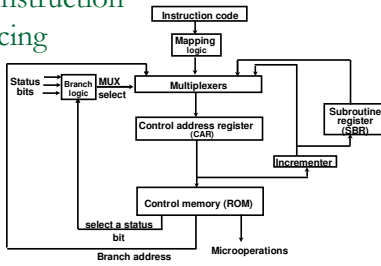
## How Does It Work?

- The next step is to generate the microoperations that execute the instruction fetched from memory
  - Depends on the operation code part of the instruction
  - Each instruction has its own microprogram routine
- Moving back and forth in the control memory requires generating control memory addresses
  - Sequencer

## Sequencer (Microprogram Sequencer)

- **Sequencer (Microprogram Sequencer)**
  - A Microprogram Control Unit that determines the Microinstruction Address to be executed in the next clock cycle
- The address sequencing capabilities require
  - Incrementing the control address register
  - Unconditional branch or conditional branch, depending on status bits conditions
  - A mapping process from the bits of the Instruction to an address for control memory
  - A facility for subroutine call and return

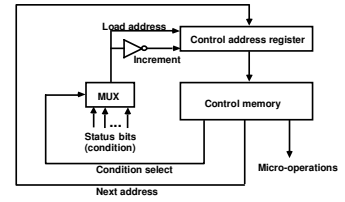
## Micro Instruction Sequencing



### Sequencing Capabilities Required in a Control Storage

- Incrementing of the control address register
- Unconditional and conditional branches
- A mapping process from the bits of the machine instruction to an address for control memory
- A facility for subroutine call and return

## Conditional Branching



### Conditional Branch

If *Condition* is true, then *Branch* (address from the next address field of the current microinstruction)

else

*Fall Through*

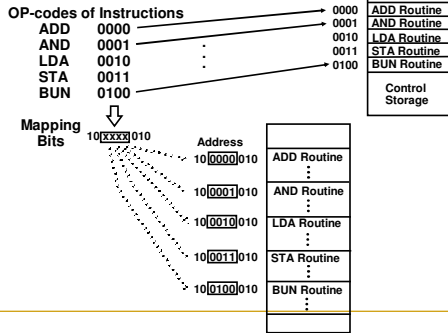
Conditions to Test: O(overflow), N(negative), Z(zero), C(carry), etc.

### Unconditional Branch

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

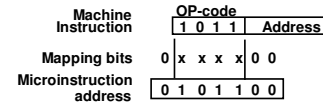
## Mapping of Instructions

### Direct Mapping

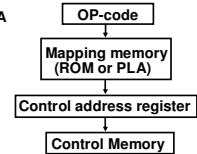


## Mapping of Instructions to Microroutines

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

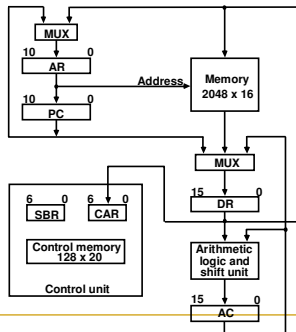


Mapping function implemented by ROM or PLA



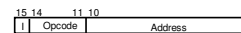
## Microprogram Example

### Computer Configuration



## Machine Instruction/Micro Instruction Formats

### Machine instruction format

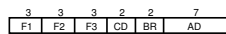


### Sample machine instructions

Symbol	OP-code	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

EA is the effective address

### Microinstruction Format



F1, F2, F3: Microoperation fields  
 CD: Condition for branching  
 BR: Branch field  
 AD: Address field

## Micro Instruction's Fields Description

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	INPC
110	PC ← AR	ARTPC
111	Reserved	

## Micro Instruction's Fields Description 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 Micro Instructions

- 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 may specify a symbolic address terminated by a colon
    - Micro-ops, consists of one, two or three symbols separated by commas
    - CD: one of {U, I, S, Z}
    - BR: one of {JMP, CALL, RET, MAP}
    - AD: one of {Symbolic Address, NEXT, empty}

## Symbolic Microprogram For 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, INPC U JMP NEXT  
DRTAR U MAP

Binary equivalents translated by an assembler

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			
BRANCH:	ORG 4	S	JMP	OVER
	NOP U JMP FETCH			
OVER:	NOP I CALL INDRCT			
	ARTPC U JMP FETCH			
STORE:	ORG 8	I	CALL	INDRCT
	NOP U JMP NEXT			
	ACTDR U JMP NEXT			
	WRITE U JMP FETCH			
EXCHANGE:	ORG 12	I	CALL	INDRCT
	NOP U JMP NEXT			
	ACTDR, DRTAC U JMP NEXT			
	WRITE U JMP FETCH			
FETCH:	ORG 64			
	PCTAR U JMP NEXT			
	READ, INPC U JMP NEXT			
	DRTAR U MAP			
INDRCT:	READ U JMP NEXT			
	DRTAR U RET			

## Binary Microprogram

Micro Routine	Address		Binary Microinstruction				AD	
	Decimal	Binary	F1	F2	F3	CD		BR
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
	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
	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000