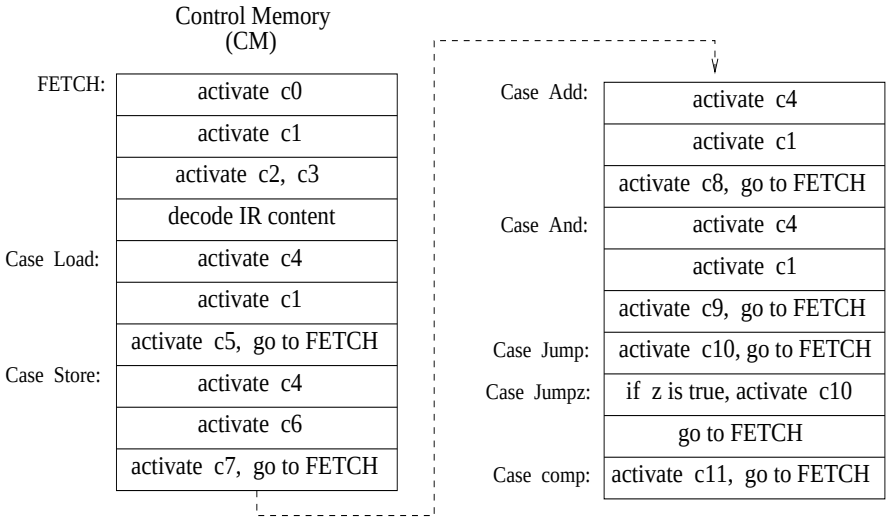
Lecture 16-17: March 19, 2021
Computer Architecture and Organization-I
Biplab K Sikdar

0.4 Example Design - Microprogrammed CU

Let consider the design of CU for the CPU with 7 macro instructions.



Microprogram for CU design

Consider FETCH is a macro-instruction. Now, assume opcode for

$FETCH$	000
$LOAD$	001
$STORE$	010
ADD	011
AND	100
$JUMP$	101
$JUMPZ$	110
$COMP$	111

Maximum number of micro-instructions required for a micro-program is 4.

For FETCH it is 3 + 1 go to IR (decode IR content).

We keep 4 words of CM per micro-program.

That is, CM is having $4 \times 8 = 32$ words.

A word then can be referred by 5-bit addresses.

We apply horizontal formatting.

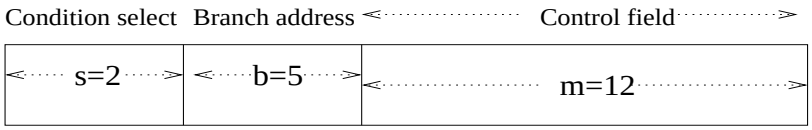
That is, control field is with 12 bits corresponding to 12 control signals.

Considering all the micro-programs - it can be found that there are 4 branch conditions - No branch, Branch always (go to FETCH), conditional branch (if Z is true) and go to IR.

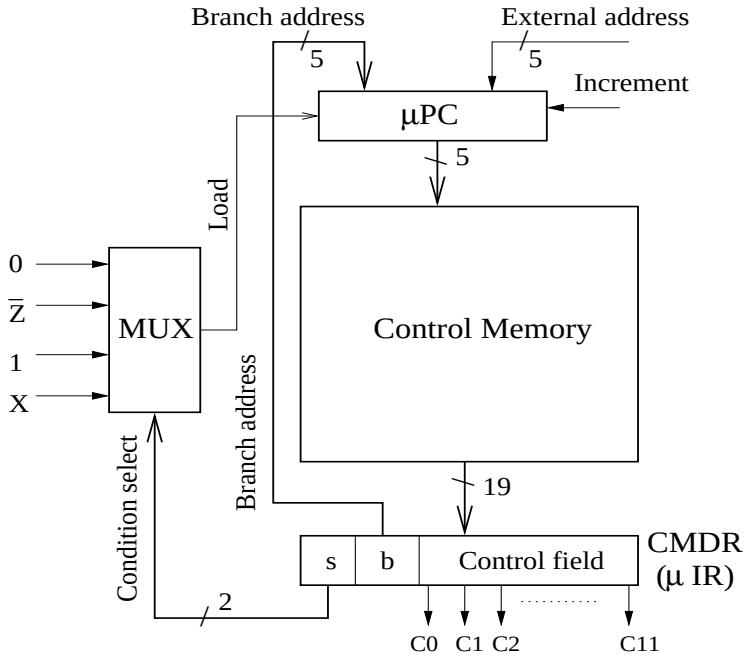
That is, condition select s = 2 bits and we need a 4 to 1 MUX[1],

That is, each word of CM is of 2 + 5 + 12 = 19 bits.

[1]When s = 11 in Figure 15, X input is passed to the output of MUX. It is assumed that this realizes go to IR (-that is, the content of IR with two augmented 0s is loaded to μPC).



a) Micro−instruction format



b) Micro−programmed control unit structure

Go to IR:

If content of IR is 110, then go to 11000 (opcode + 2 augmented 0s with LSB).

It defines address mapping: the starting address of a micro-program coresponding to a macro-instruction is ”opcode + 2 augmented 0s with LSB”.

Content of control memory

Macro	*Address*	*Condition select*	*Branch address*	$c_0c_1c_2$	$c_3c_4c_5$	$c_6c_7c_8$	$c_9c_{10}c_{11}$
FETCH	00000	00	00000	100	000	000	000
	00001	00	00000	010	000	000	000
	00010	00	00000	001	100	000	000
	00011	11	00000	*go*	*to*	*IR*	
LOAD	00100	00	00000	000	010	000	000
	00101	00	00000	010	000	000	000
	00110	10	00000	000	001	000	000
	00111						
STORE	01000	00	00000	000	010	000	000
	01001	00	00000	000	000	100	000
	01010	10	00000	000	000	010	000
	01011						
ADD	01100	00	00000	000	010	000	000
	01101	00	00000	010	000	000	000
	01110	10	00000	000	000	001	000
	01111						
AND	10000	00	00000	000	010	000	000
	10001	00	00000	010	000	000	000
	10010	10	00000	000	000	000	100
	10011						
JUMP	10100	10	00000	000	000	000	010
	10101						
	10110						
	10111						
JUMPZ	11000	01	00000	000	000	000	000
	11001	10	00000	000	000	000	010
	11010						
	11011						
COMP	11100	10	00000	000	000	000	001
	$----$						

Example: Execution of the following program segment in MM.

$$
\begin{array}{rll}
S: & LOAD & L \\
S+1: & ADD & Y \\
S+2: & JUMPZ & X \\
 & \vdots & \\
X: & ... & ... \\
 & \vdots & \\
L: & 05 & \\
 & \vdots & \\
Y: & 13 &
\end{array}
$$

1. Set PC $\leftarrow$ S, then *execute* $\Rightarrow$ μPC $\leftarrow$ 00000.

2. Location 00000 of CM is fetched - c_0 is activated $\Rightarrow$ AR $\leftarrow$ S, μPC$\leftarrow$ 00001, no branch.

3. Location 00001 of CM is fetched - c_1 is activated $\Rightarrow$ DR $\leftarrow$ 001 L, μPC$\leftarrow$ 00010, no branch.

4. Location 00010 of CM is fetched - c_2 and c_3 are activated $\Rightarrow$ PC $\leftarrow$ S+1, IR $\leftarrow$ 001, μPC$\leftarrow$ 00011, no branch.

5. Location 00011 of CM is fetched - μPC$\leftarrow$ 00100, go to IR $\Rightarrow$ μPC$\leftarrow$ 00100.

6. Location 00100 of CM is fetched - c_4 is activated $\Rightarrow$ AR $\leftarrow$ L, μPC$\leftarrow$ 00101, no branch.

7. Location 00101 of CM is fetched - c_1 is activated $\Rightarrow$ DR $\leftarrow$ 05, μPC$\leftarrow$ 00110, no branch.

8. Location 00110 of CM is fetched - c_5 is activated $\Rightarrow$ AC $\leftarrow$ 05, μPC$\leftarrow$ 00111, branch always - μPC$\leftarrow$ 00000.

9. Location 00000 of CM is fetched - c_0 is activated $\Rightarrow$ AR $\leftarrow$ S+1, μPC$\leftarrow$ 00001, no branch.

.....

Micro-programmed CU is slower than hardwired CU.

It is easy to modify/upgrade.

Example: Assume introduction of a new instruction *CLEAR* in place of ADD.

Micro-instructions to be executed for CLEAR macro-instruction are

$$\begin{aligned} &\text{DR} \leftarrow \text{AC} && < c_6 >, \\ &\text{AC} \leftarrow \text{AC'} && < c_{11} >, \\ &\text{AC} \leftarrow \text{AC} \wedge \text{DR} && < c_9 >. \end{aligned}$$

So, opcode for CLEAR is 011.

Therefore, content of CM locations 01100 to 01110 are to be modified.

$Macro$	$Address$	$Condition\ select$	$Branch\ address$	$c_0c_1c_2$	$c_3c_4c_5$	$c_6c_7c_8$	$c_9c_{10}c_{11}$
⋮							
$CLEAR$	01100	00	00000	000	000	100	000
	01101	00	00000	000	000	000	001
	01110	10	00000	000	000	000	100
	01111						
⋮							

0.5 Control Optimization

Length of each micro-program depends on

a. Complexity of micro-instruction, and

b. Parallelism of micro-operations on the available data path.

Target of control optimization is -

Reduce number of words H of CM, and number of bits W per word.

0.5.1 Reducing H

H depends on individual micro-program length.

H is reduced by including parallel micro-operations with in a single micro-instruction.

0.5.2 Minimizing W

CU encodes control signals corresponding to the micro-operations in a field/group in such a way that no two signals encoded in this field are activated simultaneously in any micro-instruction -that is, these should be mutually exclusive.

In example design, c_2 and c_3 are to be placed in two different groups.

Two control signals c_i and c_j ($i \neq j$) are called mutually exclusive (also called compatible) if $c_i \in I_k$, then $c_j \notin I_k$ for all k.

A set of control signals $C_r = \{c_i, c_j, c_k, \cdots\}$ with pairwise compatibility among $\{c_i, c_j, c_k, \cdots\}$ is a compatibility class.

All these control signals can be encoded in the same control field/group.

If number of such control signals in C_r is N_r, then number of bits in control field of a micro-instruction is

$$W = \sum_k \lceil log_2(N_r + 1) \rceil.$$ [2]

k denotes number of compatibility classes.

That is, target of reducing W effectively leads to target of reducing k.

This can be realized by increasing cardinality ($|N_r|$) of each compatibility class.

[2] +1 is to encode the don't activate a control signal option.

The following steps can be followed to reduce W.

Step 1 : Find maximal compatibility classes (MCCs) of control signals.

Step 2 : Determine all minimal set of MCCs that include each of control signals. Each of these sets is the minimal MCC cover.

Step 3 : a) Check each MCC cover to determine all ways of including each control signal exactly in one MCC (C_r).

b) Compute W of each such resulting solution.

c) Select one with minimum W.

Example 0.1 Minimize W to encode control signals of micro-instructions -

Microinstruction	*Control signals*
I_1	$c_1,\ c_2,\ c_3,\ c_7$
I_2	$c_1,\ c_3,\ c_5,\ c_8$
I_3	$c_1,\ c_4,\ c_6$
I_4	$c_2,\ c_3,\ c_6$

Step 1: Computation of the maximal compatibility classes (MCCs).

S_1: $\{\underline{c_1}, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$

S_2: $\{c_2c_4, c_2c_5, c_2c_8, \underline{c_3c_4}, c_4c_5, c_4c_7, c_4c_8, c_5c_6, c_5c_7, c_6c_7, c_6c_8, c_7c_8\}$

S_3: $\{c_2c_4c_5, c_2c_4c_8, c_4c_5c_7, c_4c_7c_8, c_5c_6c_7, c_6c_7c_8\}$

S_4: $\{\phi\}$.

The 8 MCCs are $MC_1 = \{c_1\}$, $MC_2 = \{c_3,c_4\}$, $MC_3 = \{c_2,c_4,c_5\}$, $MC_4 = \{c_2,c_4,c_8\}$, $MC_5 = \{c_4,c_5,c_7\}$, $MC_6 = \{c_4,c_7,c_8\}$, $MC_7 = \{c_5,c_6,c_7\}$ and $MC_8 = \{c_6c_7c_8\}$.

Step 2: Extract minimal MCC covers.

To identify minimal MCC covers, construct cover table (Table 1).

A row is for MC_i and column is for control signal c_j.

X entry denotes - $c_j \in MC_i$ -that is, MC_i covers c_j.

A bold **X** in an MC denotes MC is the essential MCC.

In Table 1, essential MCCs are the MC_1 and MC_2.

Table 1: Cover table

	Control signals							
MCCs	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
$MC_1 = c_1$	**X**							
$MC_2 = c_3c_4$			**X**	X				
$MC_3 = c_2c_4c_5$		X		X	X			
$MC_4 = c_2c_4c_8$		X		X				X
$MC_5 = c_4c_5c_7$				X	X		X	
$MC_6 = c_4c_7c_8$				X			X	X
$MC_7 = c_5c_6c_7$					X	X	X	
$MC_8 = c_6c_7c_8$						X	X	X

The next tasks are -

a. Delete essential MCCs (MC_1 and MC_2) from cover table - Column c_1 and c_3 are deleted.

As MC_2 includes c_4, column c_4 can also be deleted (Table 2).

	Control signals							
MCCs	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
$MC_1 = c_1$	**X**							
$MC_2 = c_3c_4$			**X**	X				
$MC_3 = c_2c_4c_5$		X		X	X			
$MC_4 = c_2c_4c_8$		X		X				X
$MC_5 = c_4c_5c_7$				X	X		X	
$MC_6 = c_4c_7c_8$				X			X	X
$MC_7 = c_5c_6c_7$					X	X	X	
$MC_8 = c_6c_7c_8$						X	X	X

Table 2: Cover table after elimination of essential rows and corresponding columns

	Control signals				
MCCs	c_2	c_5	c_6	c_7	c_8
$MC_3 = c_2c_4c_5$	X	X			
$MC_4 = c_2c_4c_8$	X				X
$MC_5 = c_4c_5c_7$		X		X	
$MC_6 = c_4c_7c_8$				X	X
$MC_7 = c_5c_6c_7$		X	X	X	
$MC_8 = c_6c_7c_8$			X	X	X

b. If cover table contains two or more identical columns, then all but one of those columns can be deleted.

In the current example, no such case.

c. Delete dominating column c_i -

c_i dominates c_j if c_i contains an X in every row where c_j contains an X and c_i contains more Xs than c_j.

Dominating column c_i is deleted as the MCC that covers c_j also covers c_i.

In the current example, c_7 dominates c_6 (Table 3).

	Control signals				
MCCs	c_2	c_5	c_6	c_7	c_8
$MC_3 = c_2c_4c_5$	X	X			
$MC_4 = c_2c_4c_8$	X				X
$MC_5 = c_4c_5c_7$		X		X	
$MC_6 = c_4c_7c_8$				X	X
$MC_7 = c_5c_6c_7$		X	X	X	
$MC_8 = c_6c_7c_8$			X	X	X

Table 3 is after eliminating dominating column.

Table 3: Cover table after elimination of dominating columns

	Control signals			
MCCs	c_2	c_5	c_6	c_8
$MC_3 = c_2c_4c_5$	X	X		
$MC_4 = c_2c_4c_8$	X			X
$MC_5 = c_4c_5c_7$		X		
$MC_6 = c_4c_7c_8$				X
$MC_7 = c_5c_6c_7$		X	X	
$MC_8 = c_6c_7c_8$			X	X

d. Delete dominated row MC_j -

MC_i dominates MC_j if MC_i contains X in every col where MC_j is having.

Dominated row MC_j is deleted since MC_i covers all control signals that are covered by MC_j.

In example, MC_3 dominates MC_5, MC_4 dominates MC_6, MC_7 dominates MC_5, and MC_8 dominates MC_6. Dominated rows are MC_5 and MC_6.

	Control signals			
MCCs	c_2	c_5	c_6	c_8
$MC_3 = c_2c_4c_5$	X	X		
$MC_4 = c_2c_4c_8$	X			X
$MC_5 = c_4c_5c_7$		X		
$MC_6 = c_4c_7c_8$				X
$MC_7 = c_5c_6c_7$		X	X	
$MC_8 = c_6c_7c_8$			X	X

Cover table after removal of dominated rows is shown in Table 4.

Table 4: Cover table after elimination of dominated rows

	Control signals			
MCCs	c_2	c_5	c_6	c_8
$MC_3 = c_2c_4c_5$	X	X		
$MC_4 = c_2c_4c_8$	X			X
$MC_7 = c_5c_6c_7$		X	X	
$MC_8 = c_6c_7c_8$			X	X

Minimal cover for Table 4 contains two MCCs and possible choices are

$\{MC_3, MC_8\}$ and $\{MC_4, MC_7\}$.

Therefore, desired MCC solutions are

$\{MC_1, MC_2, MC_3, MC_8\}$ and $\{MC_1, MC_2, MC_4, MC_7\}$.

Step 3: Find minimum cost W.

a. In minimal MCC cover $\{MC_1, MC_2, MC_3, MC_8\}$, each of the control signals c_1, c_2, c_3, c_5, c_6, c_7 and c_8 can be covered by only one MCC (Table 5).

Table 5: Cover table to find cost

	Control signals							
MCCs	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
$MC_1 = c_1$	**X**							
$MC_2 = c_3c_4$			**X**	X				
$MC_3 = c_2c_4c_5$		X		X	X			
$MC_8 = c_6c_7c_8$						X	X	X

Control signal c_4 can be covered by either MC_2 or MC_3 (Table 5).

i. Let c_4 is included in MC_2 - MCC cover include partitions as

$\{\{MC_1 = c_1\}, \{MC_2 = c_3c_4\}, \{MC_3 = c_2c_5\}, \{MC_8 = c_6c_7c_8\}\}$.

That is, 4 groups in diagonal format.

It requires 1-bit for first group MC_1 (to activate c_1 or don't activate),

2-bit for second group MC_2 (c_3c_4), 2-bit for third MC_3 (c_2c_5) and

2-bit for fourth group MC_8 ($c_6c_7c_8$).

That is, cost of control field is

$$W = 1+2+2+2 = 7.$$

ii. Let c_4 is included in MC_3 - MCC cover include partitions as $\{\{MC_1 = c_1\}, \{MC_2 = c_3\}, \{MC_3 = c_2c_4c_5\}, \{MC_8 = c_6c_7c_8\}\}$.

For this, cost is

$$W = 1+1+2+2 = 6.$$

b. Similarly, for other MCC cover $\{MC_1, MC_2, MC_4, MC_7\}$, there can be two alternative partitions.

For which cost W= 7 and 6 respectively.