CSU22022 Computer Architecture I

Nineteenth Lecture - Microprogrammed Control

Michael Manzke

2023-2024

Trinity College Dublin

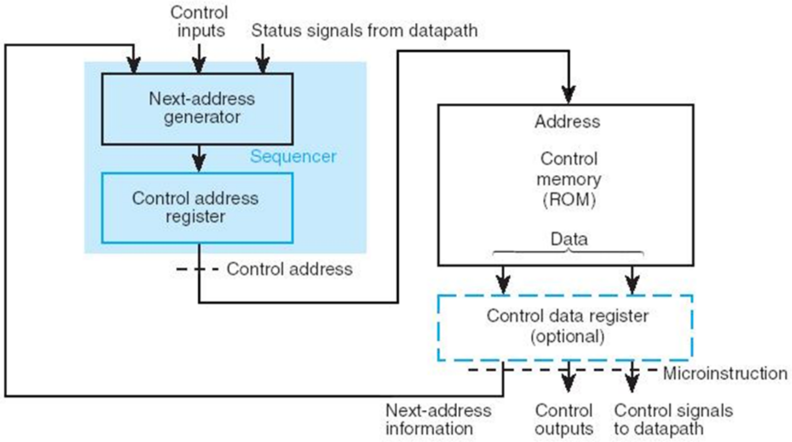
Microprogrammed Control

- Hardwired control units have the advantage of great compactness and low propagation delay.
- However, as the number of states and control signals increases, they become increasingly costly to:
  - Design
  - Debug
  - Upgrade
- A more flexible approach is used.

A Flexible Approach

- We store the control words, together with next-state information, in a control memory which is usually:
  - ROM
    - Read-only Memory
  - EPROM
    - Erasable Programmable Read-only Memory
- Then use the control inputs and status signals to select the appropriate address of the next state.
  - See figure on the next slide:

Microprogrammed Control Unit Organisation



Control Input

State		Control Input	RT	Control Word
IDLE	$\cdot$	$G = 0$	none	CW_1
IDLE	$\cdot$	$G = 1$	$C \leftarrow 0, A \leftarrow 0$	CW_2
MUL0	$\cdot$	$Q_0 = 0$	none	CW_3
MUL0	$\cdot$	$Q_0 = 1$	$A \leftarrow A + B, C \leftarrow C_{out}$	CW_4
more			more	more

Table 1: One difference which emerges is that since the control words are now stored, they cannot be made to depend dynamically on the on the value of the control input

Binary Multiplier ASM

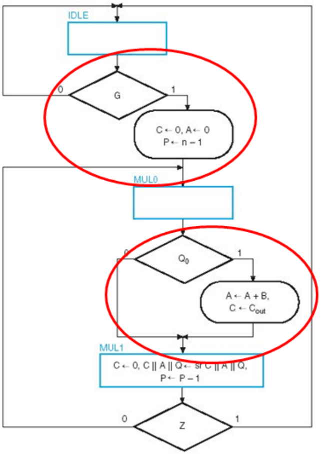


Figure 1: We must introduce additional states to supply these alternative control words.

Microprogrammed Control Unit ASM

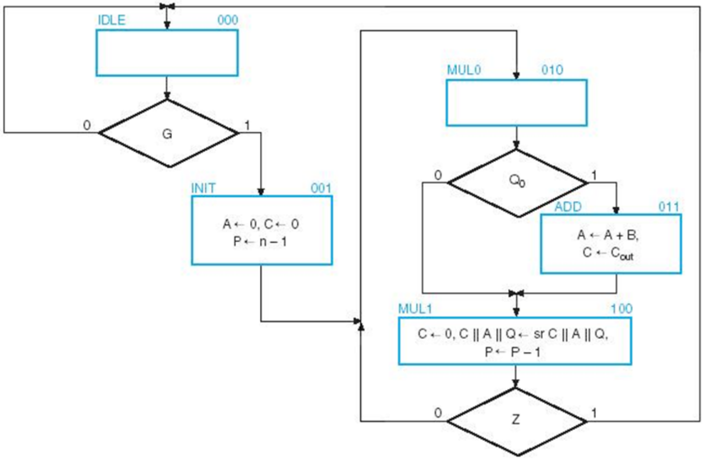


Figure 2: Two more states: INIT (001) and ADD (011)

Control Address Register (CAR)

- We have five states in the ASM, so that the control memory must store five control words:
  - We will need a 3-bit address register
    - Control Address Register (CAR)

SEL bits

- The sequence must be able to respond to two status bits:
    - Z
    - Q_0
- One control input:
    - G
- Hence two SEL bits must be included in the control word.

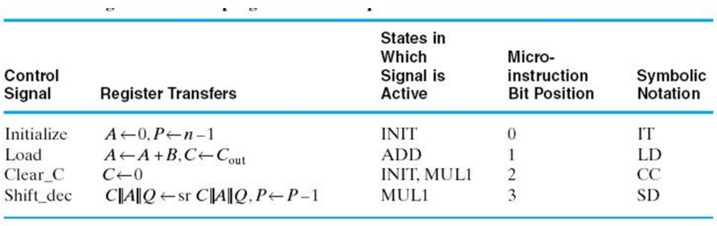
Next-Address Fields

- Finally, we add two next-address fields:
    - NXTADD0
    - NXTADD1
- This design makes no assumption about the sequence of control word accesses.
- The functional design of the sequence is as follows on the next slide:

SEL Field Definition

Symbolic notation	Binary Code	Sequencing Microoperations
NXT	00	$CAR \leftarrow NXTADD0$
DG	01	$\overline{G} : CAR \leftarrow NXTADD0$
		$G : CAR \leftarrow NXTADD1$
DQ	10	$\overline{Q_0} : CAR \leftarrow NXTADD0$
		$Q_0 : CAR \leftarrow NXTADD1$
DZ	11	$\overline{Z} : CAR \leftarrow NXTADD0$
		$Z : CAR \leftarrow NXTADD1$

Control Signals for Multiplier control



Control Signal	Register Transfers	States in Which Signal is Active	Micro-instruction Bit Position	Symbolic Notation
Initialize	$A \leftarrow 0, P \leftarrow n-1$	INIT	0	IT
Load	$A \leftarrow A+B, C \leftarrow C_{\text{out}}$	ADD	1	LD
Clear_C	$C \leftarrow 0$	INIT, MUL1	2	CC
Shift_dec	$C \| A \| Q \leftarrow \text{sr } C \| A \| Q, P \leftarrow P-1$	MUL1	3	SD

Figure 3: The control word must supply four control signals

Control Word

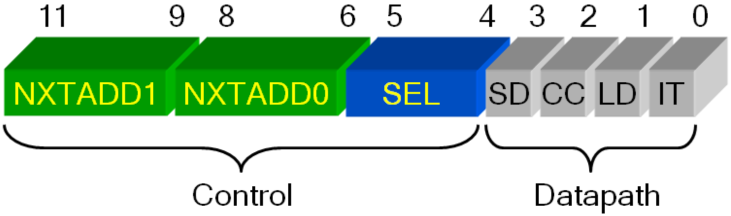
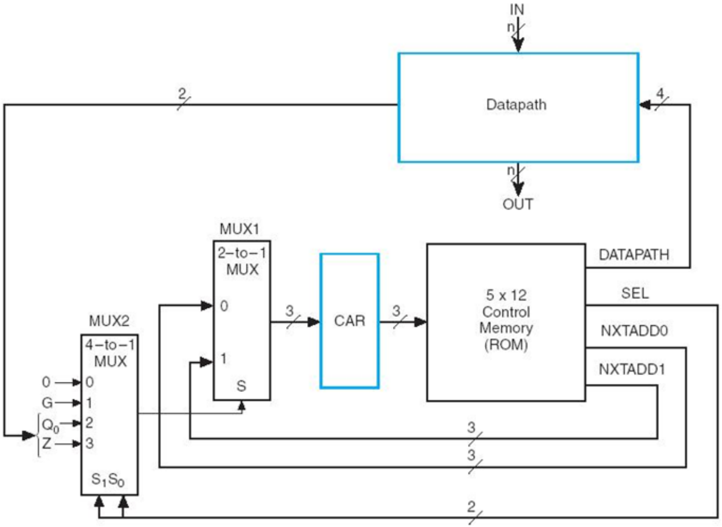


Figure 4: This results in a 12-bit control word.

Control Unit



Control Memory One

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

entity BM_Control_Memory_XXXXXXXX is
    Port ( Address : in STD_LOGIC_VECTOR (2 downto 0);
           IT : out STD_LOGIC;
           LD : out STD_LOGIC;
           CC : out STD_LOGIC;
           SD : out STD_LOGIC;
           SEL : out STD_LOGIC_VECTOR (1 downto 0);
           NXTADD0 : out STD_LOGIC_VECTOR (2 downto 0);
           NXTADD1 : out STD_LOGIC_VECTOR (2 downto 0));
end BM_Control_Memory_XXXXXXXX;
```

Listing 1: BM_Control_Memory_XXXXXXXX entity

Control Memory Two

```vhdl
architecture Behavioral of BM_Control_Memory_XXXXXXXX is

    type ROM_array is array(0 to 4) of STD_LOGIC_VECTOR (11 downto 0);

    signal ROM : ROM_array :=(
        --|11    9|8      6|5   4|   3| 2 |   1|   0| Control Memory
        --|NXTADD1|NXTADD0|SEL |SD |CC |LD |IT | Address
          "000"   &"001"  &"10"&’1’&’0’&’0’&’0’,  -- 00
          "001"   &"110"  &"01"&’0’&’1’&’0’&’0’,  -- 01
          "010"   &"010"  &"10"&’0’&’0’&’1’&’0’,  -- 02
          "011"   &"100"  &"01"&’0’&’0’&’0’&’1’,  -- 03
          "100"   &"000"  &"10"&’1’&’1’&’1’&’1’); -- 04

    signal content_at_address : STD_LOGIC_VECTOR (11 downto 0);
```

Listing 2: BM_Control_Memory_XXXXXXXX entity

Control Memory Three

```
begin

    content_at_address <= ROM(to_integer(unsigned(Address(2
    downto 0)))) after 2ns;

    NXTADD1 <= content_at_address(11 downto 9);  -- 11-9
    NXTADD0 <= content_at_address(8 downto 6);   --  8-6
    SEL <= content_at_address(5 downto 4);       --  5-4
    SD <= content_at_address(3);                 --  3
    CC <= content_at_address(2);                 --  2
    LD <= content_at_address(1);                 --  1
    IT <= content_at_address(0);                 --  0

end Behavioral;
```

Listing 3: BM_Control_Memory_XXXXXXXX entity