Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics and Science

School of Computer Science & Statistics

Integrated Computer Science Programme **Trinity Term 2016**
Year 2
Integrated Engineering
Year 3

CS2022 – Computer Architecture I

Monday 23 May 2016 **Goldsmith Hall** **14.00 – 16.00**

Dr. Michael Manzke

Instructions to Candidates:

Attempt **three** questions. All questions carry equal marks. Each question is scored out of a total of 20 marks.

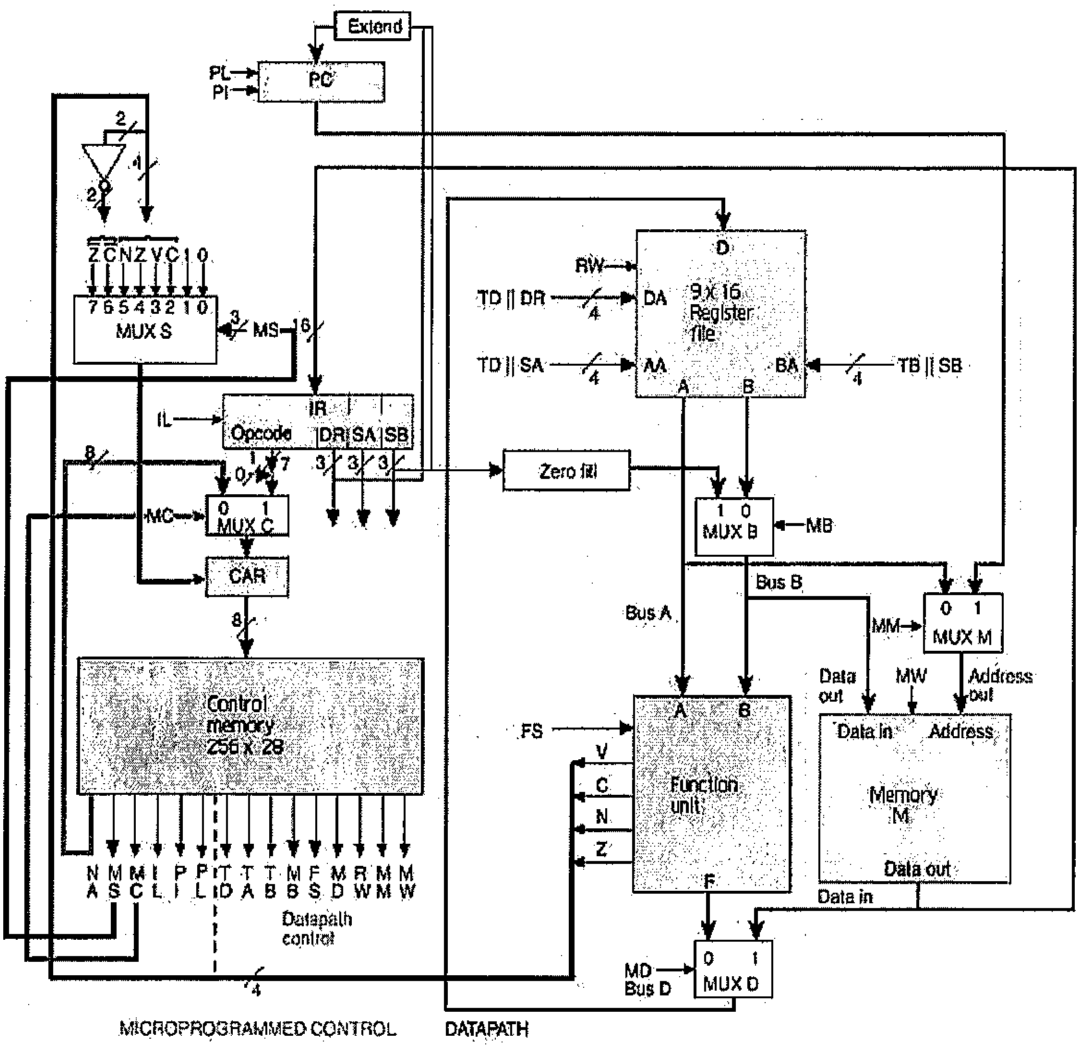
You may not start this examination until you are instructed to do so by the invigilator.

Materials Permitted for this examination:

Non-programmable calculators are permitted for this examination – please indicate the make and model of your calculator on each answer book used.

1. (a) Explain in detail the operations that take place when the following multiple cycle microprogrammed instruction set processor executes machine instructions. Your explanation must include operations in the processor's control e.g. how does one instruction in the IR register execute several control words in the Control Memory.

[15 marks]



(b) How would you micro-code a conditional branch instruction. [5 marks]

2. (a) Provide a detailed schematic for a *Function Unit* that implements the following *micro-operations*:

[15 marks]

Table 1: FS code definition

FS	Micro-operation
00000	$F = A$
00001	$F = A + 1$
00010	$F = A + B$
00011	$F = A + B + 1$
00100	$F = A + \bar{B}$
00101	$F = A + \bar{B} + 1$
00110	$F = A - 1$
00111	$F = A$
01000	$F = A \wedge B$
01010	$F = A \vee B$
01100	$F = A \oplus B$
01110	$F = \bar{A}$
10000	$F = B$
10100	$F = srB$
11000	$F = slB$

(b) How would you implement a barrel shifter? Please provide a schematic for a 4 bit barrel shifter.

[5 marks]

3. The following VHDL code implements a Binary Multiplier

```
-- VHDL code
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity binary_multiplier is
     port(CLK, RESET, G, LOADB, LOADQ: in std_logic;
       MULT_IN: in std_logic_vector(3 downto 0);
       MULT_OUT: out std_logic_vector(7 downto 0));
end binary_multiplier;

architecture behavior_4 of binary_multiplier is
     type state_type is (IDLE, MUL0, MUL1);
     signal state, next_state : state_type;
     signal A, B, Q: std_logic_vector(3 downto 0);
     signal P: std_logic_vector(1 downto 0);
     signal C, Z: std_logic;
begin
     Z <= P(1) NOR P(0);
     MULT_OUT <= A & Q;

     state_register: process (CLK, RESET)
     begin
       if (RESET = '1') then
         state <= IDLE;
       elsif (CLK’event and CLK = '1') then
         state <= next_state;
       end if;
     end process;

     next_state_func: process (G, Z, state)
     begin
       case state is
         when IDLE =>
           if G = '1' then
             next_state <= MUL0;
           else
             next_state <= IDLE;
           end if;
         when MUL0 =>
           next_state <= MUL1;
         when MUL1 =>
           if Z = '1' then
             next_state <= IDLE;
           else
             next_state <= MUL0;
           end if;
       end case;
     end process;

     datapath_func: process (CLK)
```

```
      variable CA: std_logic_vector(4 downto 0);
      begin
        if (CLK’event and CLK = '1') then
          if LOADB = '1' then
          B <= MULT_IN;
          end if;
          if LOADQ = '1' then
          Q <= MULT_IN;
          end if;
          case state is
            when IDLE =>
              if G = '1' then
                C <= '0';
                A <= "0000";
                P <= "11";
              end if;
            when MUL0 =>
              if Q(0)  = '1' then
                CA := ('0' & A) + ('0' & B);
              else
                CA := C & A;
              end if;
              C <= CA(4);
              A <= CA(3 downto 0);
            when MUL1 =>
              C <= '0';
              A <= C & A(3 downto 1);
              Q <= A(0) & Q(3 downto 1);
              P <= P - "01";
          end case;
        end if;
      end process;
end behavior_4;
```

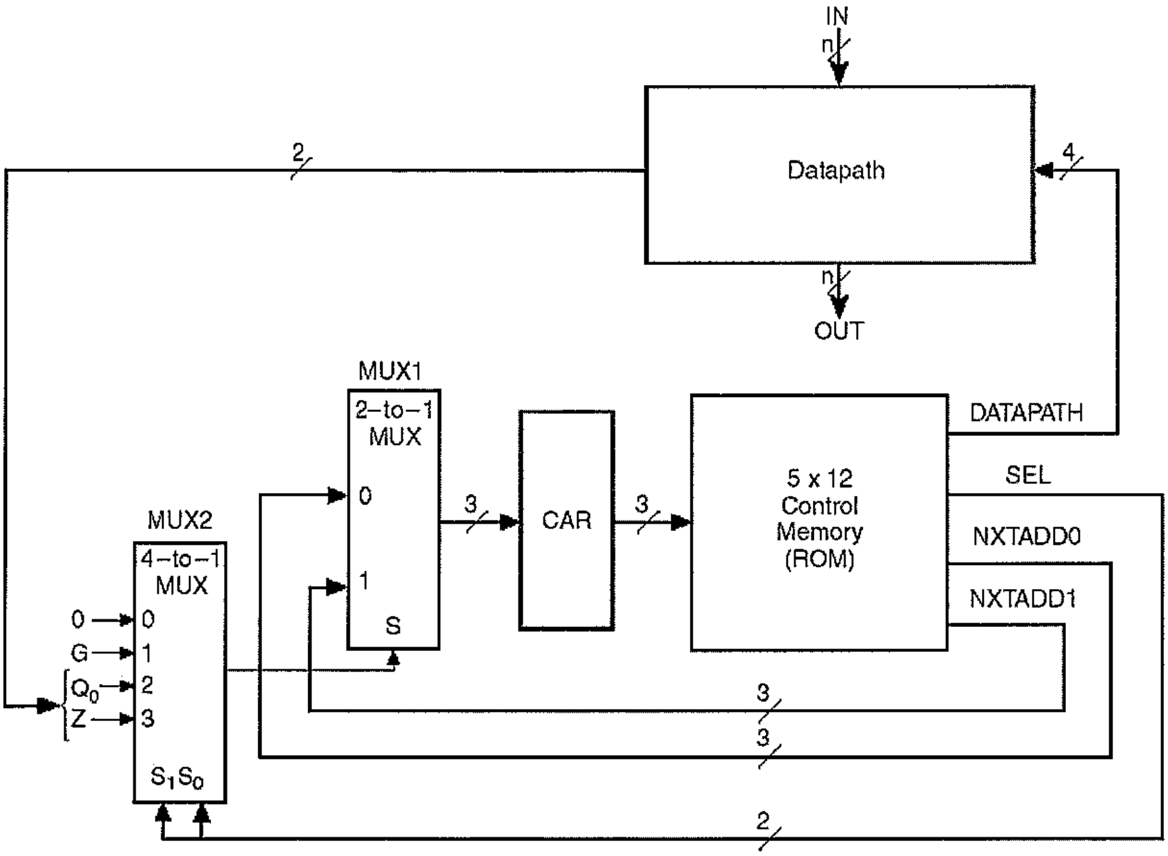
(a) Provide a schematic for the Binary Multiplier data path with a One Flip-Flop per State control unit.

[10 marks]

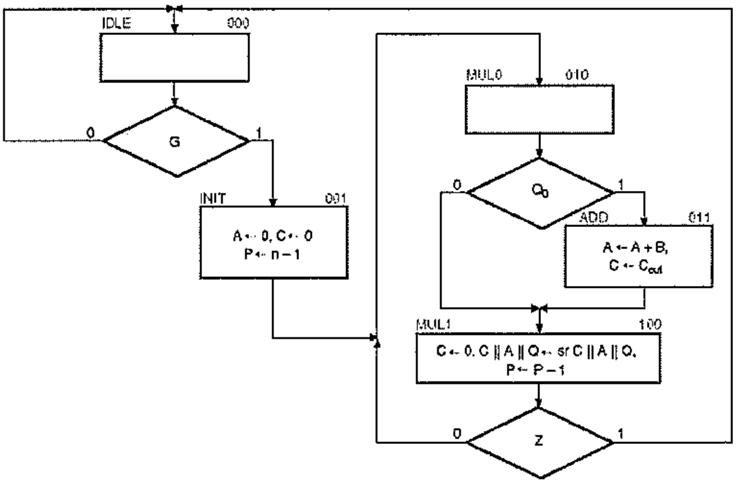
(b) Provide a schematic for the Binary Multiplier data path with a Sequence Register and Decoder control unit.

[10 marks]

4. (a) Provide micro-code for the following Microprogrammed Control Unit. This unit controls a Binary Multiplier.

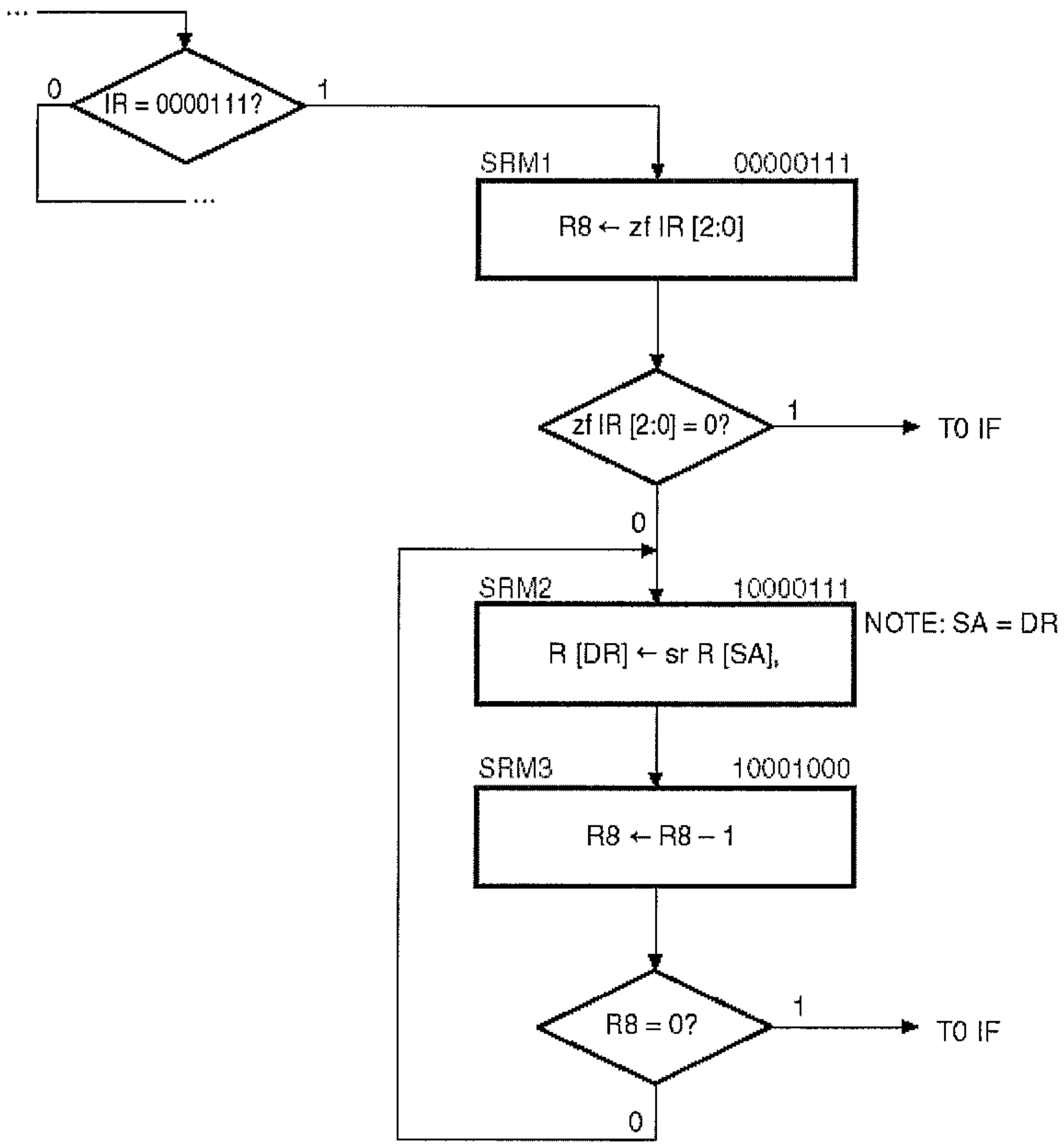


Control Signal	Register Transfers	States in Which Signal is Active	Micro-instruction Bit Position	Symbolic Notation
Initialize	$A \leftarrow 0, P \leftarrow n-1$	INIT	0	IT
Load	$A \leftarrow A + B, C \leftarrow C_{\text{out}}$	ADD	1	LD
Clear_C	$C \leftarrow 0$	INIT, MUL1	2	CC
Shift_dec	$C\|A\|Q \leftarrow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



[15 marks]

(b) The following ASM defines an instruction for our microprogrammed multiple cycle instruction set processor. What instruction does the ASM describe?



[5 marks]