UNIVERSITY OF DUBLIN

TRINITY COLLEGE

Faculty of Engineering, Mathematics and Science
School of Computer Science and Statistics

Integrated Computer Science
B.A.I in Engineering

Trinity Term 2013

CS2022 – Computer Architecture I

Tuesday 30th April 2013 Sports Centre 09:30-11:30

Dr. Michael Manzke

Answer **three** questions.

The use of non-programmable calculators is permitted.

1.

The following VHDL code implements a Binary Multiplier

```vhdl
-- 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.

[15 marks]

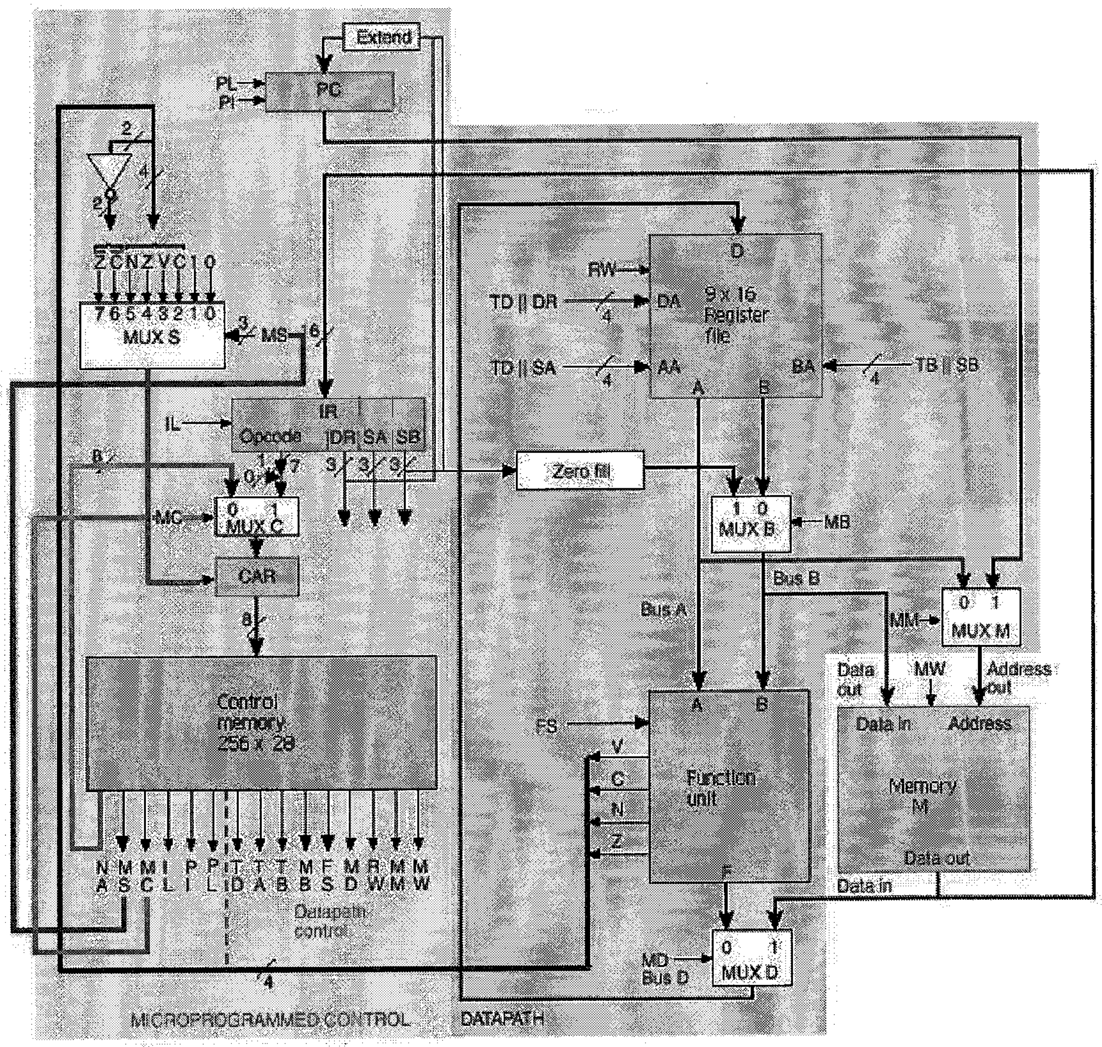
b) What modifications to the ASM chart are necessary to implement the Binary Multiplier as a microcoded control solution?

[5 marks]

2.

a) Explain in detail the operations that take place when the following multiple-cycle microprogrammed instruction set processor executes machine instructions.

[15 marks]



b) Expand on your discussion from Question 2.a) by explaining how the following instruction is executed, Immediate Instruction

[5 marks]

3.

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) What do the following Boolean Expressions implement? Please provide a detailed discussion.

$C_{i+1} = g_i + p_iC_i$

$C_1 = x_0y_0 + C_0(x_0 + y_0)$

$= g_0 + C_0p_0$

$C_2 = x_1y_1 + C_1(x_1 + y_1)$

$= x_1y_1 + [x_0y_0 + C_0(x_0 + y_0)](x_1 + y_1)$

$= g_1 + p_1g_0 + p_0p_1C_0$

$C_3 = g_2 + p_2g_1 + p_1p_2g_0 + p_0p_1p_2C_0$

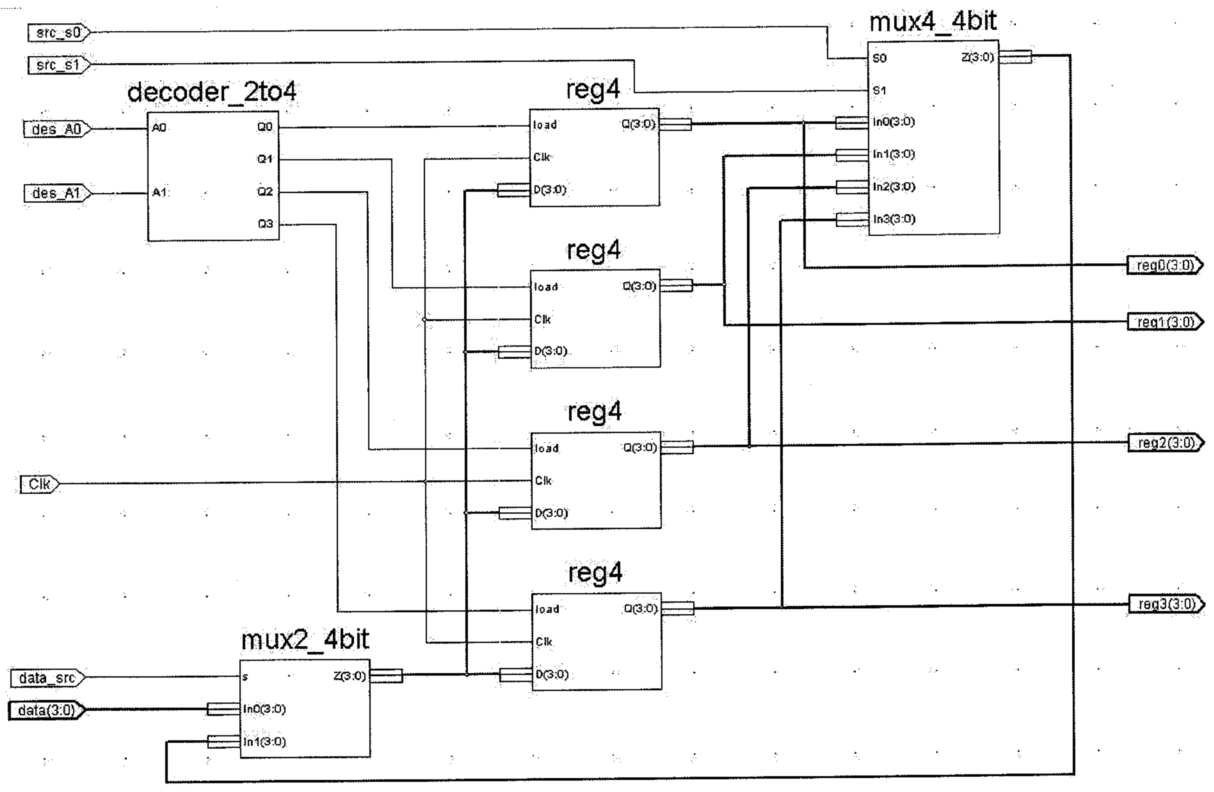
$C_4 = g_3 + p_3g_2 + p_2p_3g_1 + p_1p_2p_3g_0 + p_0p_1p_2p_3C_0$

[5 marks]

4

a) Write VHDL code that implements the following *Register-file*:

[15 marks]



b) Discuss the register transfer operations that can be performed with this Register-file.

[5 marks]

© UNIVERSITY OF DUBLIN 2013