THE AUSTRALIAN NATIONAL UNIVERSITY

First Semester Examinations 2009

ENGN3213
Digital Systems and Microprocessors

Writing Period: 3 hours
Study Period: 15 minutes duration

Permitted Materials: Pens

Total 100 Marks (30% of Subject)

You must attempt each of the 5 questions
Q1. General Knowledge (20 marks; 20% Total)

(a) Describe how the PICOBLAZE microcontroller executes a subroutine call. (3 marks)

(b) Explain using diagrams how a Schmitt Trigger can debounce a switch. (3 marks)

(c) (1) Draw the truth table of a T flip-flop. (2 marks)
      (2) Sketch a circuit implementation of the T flip-flop using one D-flip-flop. (3 marks)

(d) The D flip-flop shown in the following figure is applied the input signals shown in the traces.
     (1) Define in words the meaning of setup and hold time (2 marks)
     (2) Redraw the traces in the figure in your script books and label the setup and hold times of the flip-flop (2 marks).

```
D   Q
CLK

D
CLK
```
(e) Describe line by line the function of the following PICOBLAZE assembly language program (5 marks).

```
CONSTANT counter_port, 04
NAMEREG sA, interrupt_counter
start: LOAD interrupt_counter, 00
       ENABLE INTERRUPT
       ;
       ADDRESS 3B0
intRoutine: ADD interrupt_counter, 01
            OUTPUT interrupt_counter, counter_port
            RETURN ENABLE
            ;
            ADDRESS 3FF
            JUMP intRoutine
```
Q2. Arithmetic (20 marks; 20% Total)

Evaluate the following (2 mark each),

(a) \(129_{10} = X_2, \Rightarrow X = \)
(b) \(113_{16} = Y_5, \Rightarrow Y = \)
(c) \(0111_2 + 0111_2 = \)
(d) \(0100_2 - 0001_2 = \)
(e) \(1111_2 \times 0011_2 = \)
(f) \(1101111100_2 = X_{16} \Rightarrow X = \)
(g) \(1100111101_2 = Y_8 \Rightarrow Y = \)
(h) What is the integer value of the C variable A defined by,

```
signed char A = 131;
```

(i) Compute \(4_{10} + (-6)_{10}\) using two’s complement arithmetic.

(j) If unsigned short \(Y = X^2\), where \(X = 257\) what does \(Y\) evaluate to? You may assume that an unsigned short has a width of 16 bits.

Q3. State machines and RTL systems (20 marks; 20% Total)

A pulsed high voltage power supply is designed to discharge a bank of capacitors into a load. The power supply shown in the figure consists of a high voltage rectifier, a capacitor bank, a high current switch (Ignitron) and a load. The power supply is controlled by a synchronous state machine controller that you have to design.
The following RTL schematic shows the relationship of the controller to the supply. The power supply is a predominantly analogue system, however there are digital circuits embedded in the power supply that transmit and receive control signals to and from the controller. These digital circuits and their data form the data path. The control signals are described below.

The power supply functions as follows. The controller takes three inputs from a pair of buttons (CHARGE and FIRE) at the disposal of the operator and a HIGH TENSION READY signal from the power supply. In STANDBY mode the controller is ready to run a shot cycle which involves charging the bank and firing the ignitron. In STANDBY mode, assuming that there are no fault conditions (faults are not handled by the controller), the controller waits for the operator to press the CHARGE button. When the CHARGE button is pressed, the controller enters CHARGE mode and sends the CHARGE ENABLE (C<sub>en</sub>) signal to the power supply after which the power supply starts to charge the bank. When the bank is fully charged, the HIGH TENSION READY (HTR) signal is activated causing the controller to enter ARM mode where it waits for the operator to push the FIRE (F) button. With the FIRE button pushed the controller enters FIRE mode and sends a FIRE ENABLE (F<sub>en</sub>) signal to the power supply to trigger the ignitron. In FIRE mode the controller waits
for the discharge to terminate before returning to STANDBY mode.

(a) What are the states of the controller? (2 marks).
(b) Draw the state diagram for the controller (3 marks).
(c) Draw a truth table showing the next state logic (you may use D flip flops) (3 marks).
(d) Draw a truth table showing the output logic (3 marks).
(e) Use Karnaugh maps to minimise the logic for each of (c) and (d) (3 marks).
(f) Draw the resultant circuit schematic of the controller (3 marks).
(g) Starting from the next state and output logic tables, implement the controller in VERILOG HDL using standard coding practice (3 marks).
Q4. VERILOG and C (20 marks; 20% Total)

(1) (a) Write down VERILOG code which describes an AND gate using an ALWAYS procedural block where X and Y are the inputs and Z is the output (2 marks).

(b) Write down VERILOG code which describes a D-type flip-flop using an ALWAYS procedural block where \( D \rightarrow Q \) at the positive edge of the CLOCK, \( clk \). (2 marks)

(c) Draw a timing diagram starting at \( t = 0 \) and running for four clock periods showing the values of A,B,C and clk produced by the following VERILOG. (6 marks)

```verilog
module sequence;

reg clk;
reg A;
reg B;
reg C;

initial begin
    A = 0;
    clk = 0;
    forever #1 clk = ~clk;
end

always @(posedge clk) begin
    A <= ~A;
    B <= A;
    C <= B;
end

initial begin
    $dumpfile("sequence.vcd");
    $dumpvars;
end

initial begin:
    stopat
    #20; $finish;
end

endmodule
(2) Answer the following questions concerning the C programs shown below.

(a) What are the results of the two printf statements in the following C-program (2 marks).

(b) Explain what is happening when data is passed from the main routine to “sub” (3 marks).

```c
#include <stdio.h>

void sub( int var ) { 
    var = 1;
    printf("var = %d\n", var);
}

int main() { 
    int var;
    var = 2;
    sub( var );
    printf("var = %d\n", var);
    return 0;
}
```

(c) What are the results of the two printf statements in the following C-program (2 marks).

(d) Explain what is happening when data is passed from the main routine to “sub” in this case (3 marks).

```c
#include <stdio.h>

void sub( int *var) { 
    *var = 1;
    printf("var = %d\n", *var);
}

int main() { 
    int var;
    var = 2;
    sub( &var );
    printf("var = %d\n", var);
    return 0;
}
```
Q5. MU0 Microprocessor (20 marks; 20% Total)

(1) Consider the following MU0 machine language program.

```
0005 3006 1005 6000 7000 0004 0001
```

Using the figures on the last page answer the following.

(a) Describe in words what happens on each of the lines of the MU0 assembly language program (2 marks).

(b) What is the contents of the accumulator at the completion of execution (2 marks).

(c) Draw a timing diagram showing the contents of the program counter (PC), the instruction register (IR), the accumulator (ACC) and the CLOCK over the two clock periods where the instruction 6000 is first executed. Label the timing diagram with the trace values in hexadecimal. (3 marks).

(2) Write a MU0 machine language program to tally the numbers from 1 to 5 (3 marks).

(3) In this exercise you are to introduce a CALL/RETURN instruction pair into the MU0 command set so that subroutines can be called. Proceed as follows:

(a) To call a subroutine a stack is needed to store the contents of the PC register. Redesign the datapath of MU0 to include the stack. Describe any control inputs and extra hardware changes or additions that you think you need (2 marks).

(b) The CALL instruction must execute in two clock cycles. Draw a timing diagram showing the contents of the PC, the IR, the STACK, the ACC and the CLOCK on each of these clock cycles. Include traces of any relevant control signals (2 marks).

(c) Based on the new datapath, draw two figures showing the active bus lines during each of the CALL instruction FETCH and EXECUTE states in a similar manner to those shown in the figures at the top of the last page for the case of an ADD instruction being executed (2 marks).

(d) The RETURN instruction must also execute in two clock cycles. Draw a timing diagram showing the contents of the PC, the IR, the STACK, the ACC and the CLOCK on each of these clock cycles. Include traces of any relevant control signals (2 marks).

(e) Based on the new datapath, draw two figures showing the active bus lines during each of the RETURN instruction FETCH and EXECUTE states in a similar manner to that in part (c) (2 marks).
**ADD**

**FSM State Transition Table**

<table>
<thead>
<tr>
<th>state</th>
<th>F[2:0]</th>
<th>Next state</th>
<th>IRIns</th>
<th>PCIns</th>
<th>AccIns</th>
<th>M[1:0]</th>
<th>Xsel</th>
<th>Ysel</th>
<th>Asel</th>
<th>Run</th>
<th>Wen</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xxx</td>
<td>1 1 1</td>
<td>0 10</td>
<td>0</td>
<td>x</td>
<td>0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>000</td>
<td>0 0 0</td>
<td>1 00</td>
<td>0</td>
<td>1 1</td>
<td>1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0 0 1</td>
<td>0 xx</td>
<td>1</td>
<td>x</td>
<td>1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>0 0 0</td>
<td>1 01</td>
<td>1</td>
<td>1 1</td>
<td>1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>011</td>
<td>0 0 1</td>
<td>1 11</td>
<td>1 1</td>
<td>1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>0 0 1</td>
<td>0 00</td>
<td>0</td>
<td>x</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>101</td>
<td>0 0 N</td>
<td>0 00</td>
<td>0</td>
<td>x</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>110</td>
<td>0 0 Z</td>
<td>0 00</td>
<td>0</td>
<td>x</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>111</td>
<td>1 0 0</td>
<td>0 xx</td>
<td>0</td>
<td>x</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- N and Z are the Negative and Zero state of the Accumulator, respectively.
- (used to reduce the size of the table, as drawn)
- If a value is not going to be latched it doesn’t matter what it is!
  (e.g. ALU output for STC)
- STP operates by remaining in its evaluation state.

**MU0 Instruction Set**

<table>
<thead>
<tr>
<th>F</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LDA</td>
<td>Acc := [S]</td>
</tr>
<tr>
<td>1</td>
<td>STO</td>
<td>[S] := Acc</td>
</tr>
<tr>
<td>2</td>
<td>ADD</td>
<td>Acc := Acc + [S]</td>
</tr>
<tr>
<td>3</td>
<td>SUB</td>
<td>Acc := Acc - [S]</td>
</tr>
<tr>
<td>4</td>
<td>JMP</td>
<td>PC := S</td>
</tr>
<tr>
<td>5</td>
<td>JGE</td>
<td>If Acc &gt;= 0, PC := S</td>
</tr>
<tr>
<td>6</td>
<td>JNE</td>
<td>If Acc ≠ 0, PC := S</td>
</tr>
<tr>
<td>7</td>
<td>STP</td>
<td>Stop</td>
</tr>
</tbody>
</table>