THE AUSTRALIAN NATIONAL UNIVERSITY
First Semester Examinations 2009: SOLUTIONS

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)

Solutions:

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

When a CALL is reached, the contents of the PC are pushed on to the stack. The PC is replaced with the address of the subroutine instruction. The routine executes. At the RETURN statement, the stack is popped with the old PC value.

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

See lecture 11

(c) (1) Draw the truth table of a T flip-flop. (2 marks)

<table>
<thead>
<tr>
<th>d</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>0</td>
<td>Q</td>
<td>Q</td>
</tr>
</tbody>
</table>

(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) Setup time is the time that the input to a sequential device should be stable before the active clock edge for the output to be valid. Hold time is the time that the input to a sequential device should be stable after the active clock edge for the output to be valid.

(2) Redraw the traces in the figure in your script books and label the setup and hold times of the flip-flop (2 marks).

ENGN3213 Digital Systems and Microprocessors
(e) Describe line by line the function of the following PICOBLAZE assembly language program (5 marks).

As per line on the program.
(1) Define an output port at the register 04
(2) Call register sA the interrupt counter
(3) Load 00 into sA
(4) Enable interrupts
(5) Make compilation start with next code at 3B0
(6) ADD 01 to contents of sA.
(7) Return from interrupt routine with interrupts enabled.

ENGN3213 Digital Systems and Microprocessors
(8) Following line is interrupt branch point.
(9) Jump to line 6
Q2. Arithmetic (20 marks; 20% Total)

Evaluate the following (2 mark each),

(a) \( 129_{10} = X_2, \Rightarrow X = 10000001 \)

(b) \( 113_{16} = Y_5, \Rightarrow Y = 2100 \)

(c) \( 0111_2 + 011_2 = 1010 \)

(d) \( 0100_2 - 0001_2 = 0011 \)

(e) \( 1111_2 \times 0011_2 = 101101 \)

(f) \( 1101111100_2 = X_{16} \Rightarrow X = 37C \)

(g) \( 1100111101_2 = Y_8 \Rightarrow Y = 1475 \)

(h) What is the integer value of the C variable A defined by,

\[
\text{signed char } A = 131;
\]

\(-125\)

(i) Compute \( 4_{10} + (-6)_{10} \) using two’s complement arithmetic. \(00010\)

(j) If \( \text{unsigned short } Y = X \times X \), where \( X = 257 \) what does \( Y \) evaluate to? You may assume that an \( \text{unsigned short} \) has a width of 16 bits. \(513\)

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 following 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.

High voltage rectifier \(\rightarrow\) Capacitor bank \(\rightarrow\) Ignitron \(\rightarrow\) Load

ENGN3213 Digital Systems and Microprocessors
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 receive and transmit control signals to and from the controller. These digital circuits and their data form the data path. The control signals are described below.

![Controller Power Supply Diagram]

The operation of the power supply is as follows. In STANDBY mode the controller is ready to run a shot cycle which involves charging the bank and discharging it into the load. Assuming that there are no fault conditions (faults are not handled by the controller) the operator pushes the CHARGE (C) button to initiate the shot cycle. The controller then enters the CHARGE mode and sends a CHARGE ENABLE (C\text{en}) signal to the power supply. The power supply then charges the bank. When the bank is fully charged, a HIGH TENSION READY (HTR) input signal is sent from the power supply to the controller. At this point the controller enters the ARM mode. In ARM mode the controller waits for a FIRE instruction (F) from the operator to trigger the discharge. When the FIRE instruction is received the controller sends a FIRE ENABLE (F\text{en}) signal to the power supply to fire the ignitron. After a shot the controller returns to the STANDBY mode.
Exercises.

(a) What are the states of the controller? (2 marks). STANDBY (STBY), CHARGE (CHG), ARM (ARM) and FIRE (FIRE)

(b) Draw the state diagram for the controller (3 marks).

(c) Draw a truth table showing the next state logic (3 marks).

<table>
<thead>
<tr>
<th>State</th>
<th>C</th>
<th>F</th>
<th>HTR</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>STBY</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>STBY</td>
</tr>
<tr>
<td>STBY</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>CHG</td>
</tr>
<tr>
<td>CHG</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>CHG</td>
</tr>
<tr>
<td>CHG</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>ARM</td>
</tr>
<tr>
<td>ARM</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>ARM</td>
</tr>
<tr>
<td>ARM</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>FIRE</td>
</tr>
<tr>
<td>FIRE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>STBY</td>
</tr>
</tbody>
</table>

(d) Draw a truth table showing the output logic (3 marks).

<table>
<thead>
<tr>
<th>State</th>
<th>C</th>
<th>F</th>
<th>HTR</th>
<th>C_en</th>
<th>F_en</th>
</tr>
</thead>
<tbody>
<tr>
<td>STBY</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>STBY</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CHG</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CHG</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ARM</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ARM</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FIRE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
(e) Use Karnaugh maps to minimise the logic for each of (c) and (d) (3 marks).

Denote the state STDBY (00), CHG (01), ARM (11) and FIRE (10). We have four Karnaugh maps: two for the next state table and two for the outputs $C_{en}$ and $F_{en}$. Karnaugh map for first state bit

\[
\begin{array}{c|cc|c}
S_1 & S_2 & C & \text{HTR} \\
\hline
00 & 0 & 0 & 0 \\
01 & 0 & 1 & 1 \\
11 & 1 & 1 & 1 \\
10 & 0 & 0 & 0 \\
10 & 0 & 1 & 1 \\
11 & 1 & 1 & 1 \\
01 & 0 & 0 & 1 \\
00 & 0 & 0 & 0 \\
01 & 0 & 0 & 0 \\
11 & 1 & 1 & 1 \\
01 & 0 & 1 & 1 \\
00 & 0 & 0 & 0 \\
\end{array}
\]

Next State First Bit

\[
\begin{array}{c|cc|c|c|c|c}
S_1 & S_2 & C & F & H & T & R \\
\hline
00 & 0 & 0 & 0 & 0 & 0 & 0 \\
01 & 0 & 1 & 1 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 & 1 & 1 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 1 & 1 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 & 1 & 1 \\
01 & 0 & 0 & 1 & 1 & 1 & 1 \\
00 & 0 & 0 & 0 & 0 & 0 & 0 \\
01 & 0 & 0 & 1 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 & 1 & 1 \\
01 & 0 & 1 & 1 & 1 & 1 & 1 \\
00 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Next State Second Bit

\[
S_{1_{\text{next}}} = S_1 \cdot S_2 + S_2 \cdot \text{HTR}, \quad S_{2_{\text{next}}} = \overline{S_1} \cdot (C + S_2 \cdot \overline{C}) + S_2 \cdot \overline{F}
\]

\[
\begin{array}{c|cc|c|c|c|c}
S_1 & S_2 & C & \text{Fen} & \text{HTR} \\
\hline
00 & 0 & 0 & 0 & 0 \\
01 & 0 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 \\
10 & 0 & 0 & 0 & 0 \\
10 & 0 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 \\
01 & 0 & 0 & 1 & 1 \\
00 & 0 & 0 & 0 & 0 \\
01 & 0 & 0 & 0 & 0 \\
11 & 1 & 1 & 1 & 1 \\
01 & 0 & 1 & 1 & 1 \\
00 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Charge enable, $C_{en}$

\[
\begin{array}{c|cc|c|c|c|c}
S_1 & S_2 & C & \text{Fen} & \text{HTR} \\
\hline
00 & 0 & 0 & 0 & 0 \\
01 & 0 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 \\
10 & 0 & 0 & 0 & 0 \\
10 & 0 & 1 & 1 & 1 \\
11 & 1 & 1 & 1 & 1 \\
01 & 0 & 0 & 1 & 1 \\
00 & 0 & 0 & 0 & 0 \\
01 & 0 & 0 & 0 & 0 \\
11 & 1 & 1 & 1 & 1 \\
01 & 0 & 1 & 1 & 1 \\
00 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Fire enable, $F_{en}$

\[
C_{en} = \overline{S_1} \cdot S_2, \quad F_{en} = S_1 \cdot \overline{S_2}
\]
(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).

```verilog
module pps(clk, C, F, HTR, Cen, Fen);

input clk;
input C;
input F;
input HTR;

output Cen;
output Fen;

reg Cen;
reg Fen;

reg [1:0] state;
reg [1:0] nextstate;

parameter STDBY = 2'b00;
parameter CHG = 2'b01;
parameter ARM = 2'b11;
parameter FIRE = 2'b10;

always @posedge clk) state <= nextstate;

// Next state logic
always @(C or F or HTR or state) begin
    case(state)
        STDBY: if(C==1) nextstate = CHG;
               else nextstate = STDBY;
        CHG: if(HTR==1) nextstate = ARM;
             else nextstate = CHG;
        ARM: if(F==1) nextstate = FIRE;
             else nextstate = ARM;
        FIRE: nextstate = STDBY;
        default: nextstate = STDBY;
    endcase
end

// Next state logic
always @(state)
    case(state)
        STDBY: begin Cen = 0; Fen = 0; end
        CHG: begin Cen = 1; Fen = 0; end
        ARM: begin Cen = 0; Fen = 0; end
        FIRE: begin Cen = 0; Fen = 1; end
        default: begin Cen = 0; Fen = 0; end
    endcase
endmodule
```

ENGN3213 Digital Systems and Microprocessors
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).
always @(X or Y) begin
    Z = X&Y;
end

(b) Write down VERILOG code which describes a D-type flip-flop using an AL-
WAYS procedural block where $D \rightarrow Q$ at the positive edge of the CLOCK, $clk$. (2 marks)
always @(posedge clk) begin
    Q <= D;
end

(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)
(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).
   var = 1
   var = 2

(b) Explain what is happening when data is passed from the main routine to “sub” (3 marks).
The variable var is passed to sub by value. The value of var in sub is passed as 2 but then set to 1. The value var = 1 is printed. The value of var in sub is limited to the scope of sub. In main the value of var is still 2. The value var = 2 is then printed in main.

```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).
   var = 1
   var = 1

(d) Explain what is happening when data is passed from the main routine to “sub” in this case (3 marks).
The variable var is passed to sub by reference: i.e. its memory location is passed to sub. Inside sub the value stored at the memory location of var is set to 1. Then var = 1 is printed. Since the memory location is absolute, the value stored in var is still 1 after
control returns to main. The value var = 1 is again printed.

```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;
}
```

ENGN3213 Digital Systems and Microprocessors
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).
   * 0005: Load the contents of address 5 into acc
   * 3006: Subtract the contents of address 6 from the contents of the acc
   * 1005: Store the contents of the acc in address 5
   * 6000: If the contents of the acc is not zero jump back to address 0
   * 7000: stop

(b) What is the contents of the accumulator at the completion of execution (2 marks).
   000

(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).

0008
2009
200A
200B
200C
200D
1008
7000
0000
0001
0002
0003
0004
0005

ENGN3213 Digital Systems and Microprocessors
(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).

The additions shown in the above figure include a STACK to store the current value of the PC when the CALL branch is executed. To do this the input to the stack must be the PC output. The output of the STACK connects to one input port of YMUX so that the POPPED value can be sent to the PC via the ALU. The YMUX now has three inputs to multiplex the D-bus, the IR value and the STACK onto the Y-bus. The STACK has an enable and a pop/push switch input to control its function.

ENGN3213 Digital Systems and Microprocessors
(b) The CALL instruction must execute in two clock cycles. Draw a timing diagram showing the contents of the PC, the IR, the stack and the ACC on each of these clock cycles. Include traces of any relevant control signals (2 marks).
(c) Illustrate on your new datapath the active bus-lines during the CALL instruction in a similar manner to those shown in the figure at the top of the last page that shows the active paths when an ADD instruction is 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 and the ACC on each of these clock cycles. Include traces of any relevant control signals (2 marks).
(e) Illustrate on your new datapath the active bus-lines during the RETURN instruction in a similar manner to those shown in the figure at the top of the last page that shows the active paths when an ADD instruction is executed (2 marks).
**MU0 Instruction Set**

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

ENGN3213 Digital Systems and Microprocessors