Exam #:

EECS 373 Practice Midterm / Homework #3

Fall 2014

Name: _____ Uniquename: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Problem #	Points
1	/10
2	/10
3	/15
4	/20
5	/15
6	/10
7	/10
8	/10
Total	**/100**

NOTES:

- **PUT YOU NAME/UNIQUENAME ON EVERY PAGE TO ENSURE CREDIT!**
- Can refer to the ARM Assembly Quick Ref. Guide and 1 page front/back cheat sheet
- Can use a basic/scientific calculator (but not a phone, PDA, or computer)
- Don't spend too much time on any one problem.
- You have 80 minutes for the exam.
- The exam is 9 pages long, including the cover sheet.
- Show your work and explain what you are doing. Partial credit w/o this is rare.

Name: _____ Uname: _____

1) Fill-in-the-blank or circle the *best* answer. **[10 pts, all or no points for each question]**

   a) The ARM EABI specifies that the first parameter of a function call should be stored in register ____ and the return address is stored in register ____.

   b) The ARM Thumb-2 instruction set has ***16-bit / 32-bit / 16- and 32-bit*** encodings.

   c) A 40 MHz clock with a 20% duty cycle is high for _____ ***ns*** per cycle.

   d) An ARM EABI-compliant procedure that calls another procedure should ***always / sometimes / never*** save and restore the `lr` register.

   e) The SPI bus **does / does not** support flow control using the ACK bit.

   f) A ***UART / SPI / I2C*** is an asynchronous communications interface.

   g) I2C bus transfers are ***asynchronous / synchronous*** and use ***dedicated chip select lines / addresses embedded in frames***.

   h) In the Verilog hardware description language, an `@(posedge clock)` block would be used when implementing a ***mux / adder / flip-flop / priority encoder***.

   i) If multiple devices share a single interrupt line and generate interrupts at the same rate, then processor workload ***remains constant / grows linearly / grows quadratically*** with the number of devices.

   j) The RESET vector and initial stack pointer goes in the ***.text / .data / .bss*** section.

Name: _____ Uname: _____

2) Assembly, C, and the ABI. Consider the following program. **[10]**

```
/**
 * Print an int.
 */
void printint(int a) { printf("%d\n", a); }

/**
 * Sum the first n elements of array a.
 */
int add(int a[], int n) {
  int i, sum=0;
  for (i = 0; i < n; i++)
    sum+=a[i];
  printint(sum);
  return(sum);
}

/**
 * Test program.  Should print out 10.
 */
main () {
  int x[] = {1, 2, 3, 4, 5};
  add(x, 4);
}
```

Rewrite only the `add` function in ARM assembly, and make sure that your code conforms to the EABI for both the callee part (saving the registers, handing input parameters, etc.) and the caller part (passing parameters for the `printint` call, etc.). Do not use recursion to implement your code. The assembly code should assemble without errors. Write clearly and comment the code.

Name: _____ Uname: _____

3) ARM Assembly Language. **[15]**

a) Assume you wrote the ARM assembly language instruction: **`add r9, r9, r11`**
What is the machine encoding of this instruction in hex? Use the table below. **[5]**

Encoding T2 All versions of the Thumb ISA.

ADD<c> <Rdn>,<Rm>

15	14	13	12	11	10	9	8	7	6 5 4	3 2 1	0
0	1	0	0	0	1	0	0	DN	Rm	Rdn	

```
if (DN:Rdn) == '1101' || Rm == '1101' then SEE ADD (SP plus register);
d = UInt(DN:Rdn);  n = UInt(DN:Rdn);  m = UInt(Rm);  setflags = FALSE;
(shift_t, shift_n) = (SRType_LSL, 0);
if d == 15 && InITBlock() && !LastInITBlock() then UNPREDICTABLE;
if d == 15 && m == 15 then UNPREDICTABLE;
```

Machine Code: _____

b) Assume that **`r3=0xaabbccdd`** and **`r1=0x00001000`**, and all other registers and memory are initialized to zero. After running the following code, what are the values of registers **`r1`**, **`r3`**, and **`r5`**? Annotate each line of code to show your partial results. **[5]**

```
str  r3, [r1, #1]
ldrb r5, [r1], #2
orr  r5, r5, #0x0f
strh r3, [r1, #-4]!
ldr  r3, [r1]
```

`r1` = _____
`r3` = _____
`r5` = _____

c) Convert the following 16-bit, two's complement hex numbers into signed decimal. **[5]**

Hex	Decimal
0x1000	_____
0xfffc	_____
0xffff	_____

Name: _____ Uname: _____

4) Memory-Mapped I/O. **[20]**

a) Assume you have a memory-mapped register location REG_FOO on a 32-bit architecture. Modify main to add 7 to REG_FOO's value. Your code should compile without any warnings. **[10]**

```
#include <stdio.h>
#include <inttypes.h>

#define REG_FOO 0x40000140

main () {
  // Your code here to declare reg and
  // add 7 to REG_FOO's value.




  printf("0x%x\n", *reg); // Prints out new value
}
```

b) Sketch the glue logic needed to interface a D flip-flop (DFF) to the APB. The PSEL line is the peripheral select (i.e. it goes high when the processor is addressing the DFF). The DFF will be attached to data bits zero (PWDATA[0] and PRDATA[0]), perhaps through tri-state drivers. You should be able to change the value of the DFF by writing the memory location corresponding to the PSEL and read the current value of the DFFs by reading the location. Assume that the DFF has an enable (ENA) line that, when not asserted, ignores any inputs (e.g. D and CLK) but continues to drive Q and Q#. The APB signals are: PCLK, PADDR, PWRITE, PSEL, PENABLE, PWDATA, PRDATA. **[10]**

Name: _____ Uname: _____

5) Serial Buses. **[15]**

a) I2C. The standard-mode I2C specification uses open-drain logic. Under this scheme, bus devices drive the SCL and SDA bus lines low (to 0 V) but rely on a shared, external pull-up resistor (typically 10 kΩ) to send a high bit (at 3.3V or 5V) by "pulling-up" the bus line. **How long** will a low-to-high transition take, after the SDA line has been at 0V for a long time, if low is 0 V, VCC is 3 V, a "high" is 80% of VCC (e.g. 2.4 V), the aggregate bus capacitance is 50 pF, and a 10 kΩ pull-up resistor is used? Show your work. **[8]**

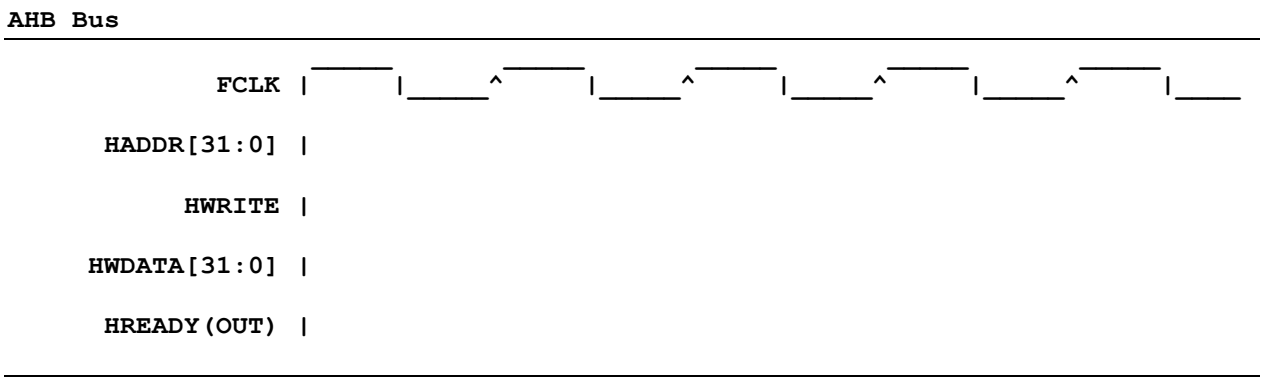
b) Assume you have four devices connected to a microcontroller over a shared I2C bus that is running at 400 kHz and using standard addressing. If the microcontroller reads one data byte from each of the four devices sequentially, what is the maximum data transfer rate in bits/second possible from each device? You analysis should report the useful data that are delivered per unit time. Note that "time" includes both the time required to send useful data and the time required for communication overhead (e.g. any start/stop/ack bits, addressing bytes, etc.). Show your work. **[7]**

Name: _____ Uname: _____

6) Logic Design. Design a digital circuit that takes an input clock of 100 MHz clock (CLKIN) and converts it into an output clock of 5 MHz with a 30% duty cycle (CLKOUT). You may use synchronously resettable D flip-flops and *n*-bit binary counters (make sure to specify the value of *n*). You may express combinational logic as Boolean expressions or using standard logic gates. Label things and write neatly. **[10]**

Name: _____ Uname: _____

7) Say your code writes the value `0x76543210` to memory location `0x20000604`, on an ARM Cortex-M3. Assume this memory location sits in an SRAM with no access delay (i.e. a zero wait state). If you could attach a logic analyzer to the AHB bus, draw a timing diagram that shows what you would expect to see on the logic analyzer for this write operation. Make sure the actual values being transferred on the bus are clearly labeled. **[10]**



Name: _____ Uname: _____

8) Write a C function **`void enable_interrupts(int x)`** that enables interrupt **`x`**. You need not check to validate that **`x`** is a legal interrupt number. The table below might be useful. **[10]**

Table 6-1 NVIC registers

Address	Name	Type	Reset	Description
0xE000E004	ICTR	RO	-	*Interrupt Controller Type Register, ICTR*
0xE000E100 - 0xE000E11C	NVIC_ISER0 - NVIC_ISER7	RW	0x00000000	Interrupt Set-Enable Registers
0xE000E180 - 0E000xE19C	NVIC_ICER0 - NVIC_ICER7	RW	0x00000000	Interrupt Clear-Enable Registers
0xE000E200 - 0xE000E21C	NVIC_ISPR0 - NVIC_ISPR7	RW	0x00000000	Interrupt Set-Pending Registers
0xE000E280 - 0xE000E29C	NVIC_ICPR0 - NVIC_ICPR7	RW	0x00000000	Interrupt Clear-Pending Registers
0xE000E300 - 0xE000E31C	NVIC_IABR0 - NVIC_IABR7	RO	0x00000000	Interrupt Active Bit Register
0xE000E400 - 0xE000E4EC	NVIC_IPR0 - NVIC_IPR59	RW	0x00000000	Interrupt Priority Register

```
void enable_interrupts(int x) {

}
```