CMPE 364 (L52)
Final Exam
Spring 2017
June 11th, 2017

Instructions: Read carefully through the entire exam first, and plan your time accordingly. Note the relative weights of each segment. Make sure that you have all pages of the exam. If you are missing any pages, inform the instructor immediately.

During this exam you may use the book *Introduction to Microprocessor Based Systems Using the Arm Processor* by Kris Schindler as well as any notes handwritten by you. The use of any other reference material will result in the confiscation of your exam and you receiving a score of 0. No electronic devices are allowed.

Write your answers on this exam. **Do not write on the back of any pages.** If you need additional space, use the extra pages at the end of the exam. Anything written on the backside of a page will not be graded.

When answering questions that require calculations, be sure to show your work and give your answer in a simplified form. Answers without appropriate work will not receive credit. When answering questions that request an explanation, keep your explanation short and correct. Explanations containing incorrect or irrelevant information will be marked wrong, even if correct information is also included.

When you are done, present your completed exam to the instructor at the head table. If leaving before the exam period is concluded, please leave as quietly as possible as a courtesy to your neighbors.

TL;DR:

- **You can use the book and handwritten notes, but nothing else.**
- **Don't write on the back of any pages, use the extra pages at the end.**
- **Simplify your answers to math problems.**
- **Show your work or you won't get points.**
- **Keep justifications concise and correct.**

Name:

Student ID Number:

Signature:

1. Basic Assembly

   (a) (4 points) What is the value of register r3 after the following instruction has executed? Assume that r1 and r2 contain `0x4ed3bfb4` and `0x8ff64da9` respectively.

   ```
   ADD r3, r2, r1
   ```

   Solution:
   0xDECA0D5D

   (b) (4 points) Write a single assembly language instruction equivalent to the following:

   ```
   ADD r1, r1, #4
   LDR r0, [r1]
   ```

   Solution: LDR r0, [r1, #4]!

   (c) (5 points) Write a short program which implements the following line of high level language pseudocode. Assume all source registers may be used as scratch registers. You may not use any multiply instructions. You may only use 4 registers. Only programs of six or less instructions will receive full points. (Longer programs that are correct will still receive the majority of the points.)

   ```
   r0 := ((r1 * 257) / 16 + (r2 + 78) / 32 + r3 / 8) / 2
   ```

   Solution:
   There are many possible solutions. The following is one:

   ```
   ADD r1, r1, r1, LSL #8
   MOV r1, r1, LSR #5
   ADD r2, r2, #78
   ADD r0 ,r1, r2, ASR #6
   ADD r0, r0, r3, ASR #4
   ```

2. Memory

(a) (5 points) Consider the following map of memory. What are the contents of registers r2 and r4 after the execution of the following instruction? Assume that r2 and r4 initially contain `0x4f790786` and `0x80084020` respectively. Also assume that processor is configured in little-endian mode.

```
LDR r2, [r4], #4
```

Address	Contents
0x80084024	0xda
0x80084023	0x65
0x80084022	0x1e
0x80084021	0x6b
0x80084020	0x28
0x8008401F	0x06
0x8008401E	0x3d
0x8008401D	0x92

Solution:

`r2 = 0x651e6b28`

`r4 = 0x80084024`

(b) (5 points) Consider the following map of memory. Determine the contents of registers r2, r3, r4, and `sp` after the execution of the following instruction, given an initial value of `0x80084018` for `sp`.

```
LDMFD sp!, {r4-r2}
```

Address	Contents
0x80084024	0xaa3333aa
0x80084020	0x99555599
0x8008401C	0x88dddd88
0x80084018	0x66000066
0x80084014	0x88222288
0x80084010	0x77111177
0x8008400C	0x44dddd44

Solution:

`r4 = 0x99555599`

`r3 = 0x88dddd88`

`r2 = 0x66000066`

`sp = 0x80084024`

3. Serial Communication

(a) (3 points) Communicating with an I_2C device requires that you know the device's address. In SPI, however, devices don't have addresses. Why do devices using I_2C need addresses while devices using SPI don't?

Solution: I_2C is designed to have more than one device on the bus at a time, so there needs to be a way for the master to specify which slave it wants to talk to. The device address is that way. In SPI, you either only have one other device on the line, or you need to use some sort of chip-select mechanism.

(b) (3 points) When communicating over UART, the two devices do not share a common clock. Given this, how do they synchronize their serial reads and writes?

Solution: Both sides need to be configured to use the same timing standard for their clocks. (Frequently 9600 bps).

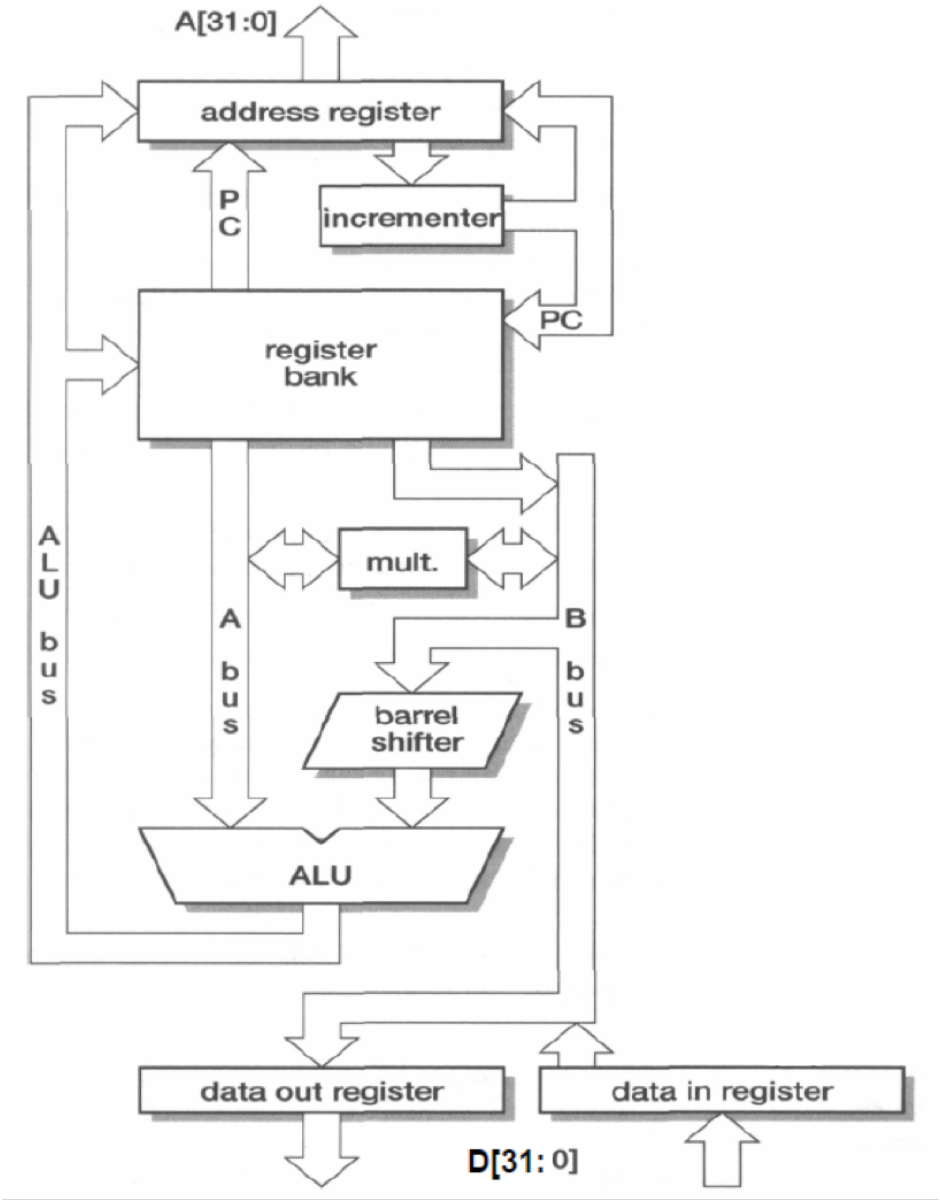
(c) (3 points) One of the configurable options when using a UART is parity. In this instance, what is parity and what does it do?

Solution: The parity bit is an extra bit sent that is used to detect errors in the transmission.

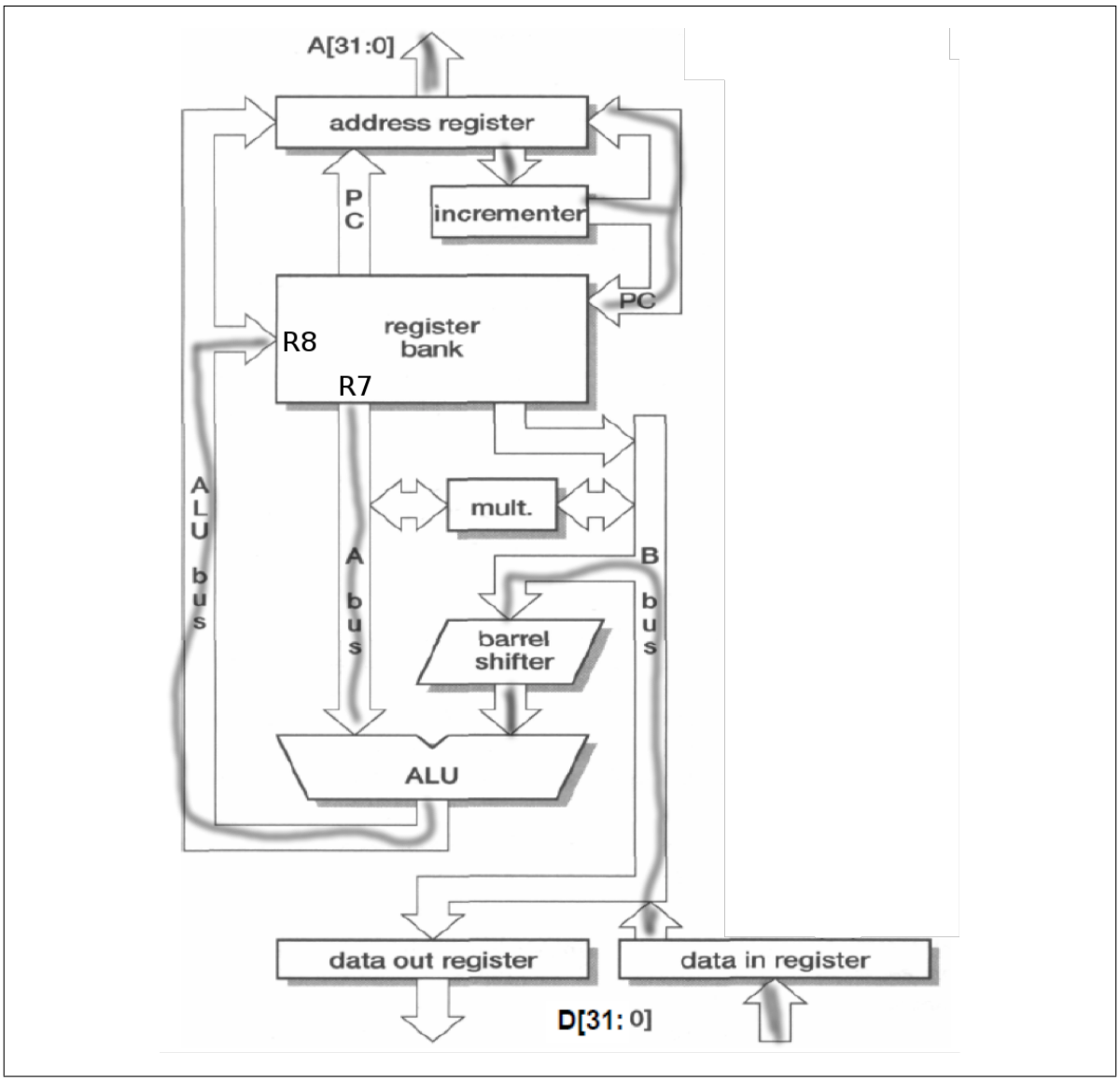
4. (6 points) Datapath

On the datapath diagram below, *shade in the portions of the datapath* used when the following instruction executes. Also, *label the read and write ports* on the register bank with which register is being outputted on that port.

```
ADD R8, R7, #8
```



Solution:


A[31:0]
address register
incrementer
PC
register bank
R8
R7
PC
ALU bus
A bus
B bus
mult.
barrel shifter
ALU
data out register
data in register
D[31: 0]

5. (10 points) Trial Subtraction

   Compute both the *quotient and remainder* of $1535 \div 9$ using trial subtraction in 8-bits. Remember that you must show all work. In addition, explicitly *label your final answer.*

Solution:

$9 * 2^7 < 1535$ $[1535 - 9 * 2^7 = 383]$

$9 * 2^6 > 383$

$9 * 2^5 < 383$ $[383 - 9 * 2^5 = 95]$

$9 * 2^4 > 95$

$9 * 2^3 < 95$ $[95 - 9 * 2^3 = 23]$

$9 * 2^2 > 23$

$9 * 2^1 < 23$ $[23 - 9 * 2^1 = 5]$

$9 * 2^0 > 5$

Therefore...

$q = 0b10101010 = 0xAA = 170$

$r = 5$

6. Diagramming Pipeline Stalls

Consider the following lines of assembly code:

```
sub r4, r4, #4
ldrb r0, [r4]
add r3, r3, r4
eor r5, r5, r0
mov r6, #45
```

(a) (8 points) *Fill in the following diagram* showing the progression of the code through the pipeline. *If there are any stalls, circle them.* You should complete the diagram until there are no more instructions in the pipeline.

Cycle #	Fetch	Decode	ALU	LS1	LS2

Solution:

Cycle #	Fetch	Decode	ALU	LS1	LS2
1	sub				
2	ldrb	sub			
3	add	ldrb	sub		
4	eor	add	ldrb	sub	
5	mov	eor	add	ldrb	sub
6		mov	eor	add	ldrb
7		mov	eor		add
8			mov	eor	
9				mov	eor
10					mov

(b) (2 points) How many cycles does it take to fully execute these instructions?

Solution: 10 cycles

7. Analog Sensors

Imagine you are interfacing a REV-11-1107-DS-00 analog pressure sensor to a microprocessor that has a 10-bit ADC unit. Both your microprocessor and the sensor use a V_{cc} of 5 volts. The datasheet for the REV-11-1107-DS-00 is included as a addendum to this exam. When reading the datasheet, ignore the part about normalizing voltage output.

The ADC is configured as follows:

- $V_{ref+} = 5V$
- $V_{ref-} = 0V$

(a) (6 points) After connecting the pressure sensor and interfacing to it with the ADC, you perform a reading and receive the following 10-bit value: `1000000000`.

What is the analog voltage being output by the sensor?

Solution:
$\frac{5V-0V}{2^{10}} * 2^9 = 2.5V$

(b) (6 points) What is the pressure that corresponds to this voltage? (Be sure to include units on your answer.)

Solution:
$p = 250 * \frac{V_{out}}{V_{cc}} - 25$
$p = 250 * \frac{2.5V}{5V} - 25$
$p = 100$ PSI

8. (15 points) Optimization

Consider the following assembly subroutine:

```
; The Doubler: A simple subroutine to double the value of every integer in an array.
; r1 contains the address of any array of integers
; r0 contains the number of items in the array
doubler:
    ; Counter for the loop
    mov r2, #0

    dloop:
        cmp r2, r0
        bge end
        ldr r3, [r1]
        mov r3, r3, LSL #1
        str r3, [r1], #+4
        add r2, r2, #1
        b dloop
    end:
        bx lr
```

Write an optimized version of this subroutine. The more optimized your solution, the more points you will receive. (Note, however, that code that doesn't work properly will receive low points, regardless of how optimal it is. So ensure your program functions properly)

You should also write a few sentences summarizing the optimizations you did.

Nothing written on this page will be graded, write your answer on the next page.

Answer for Question 8.

Solution:

```
; The Doubler: A simple subroutine to double the value of every integer in an array.
; r1 contains the address of any array of integers
; r0 contains the number of items in the array
doubler:
    ldr r3, [r1]
    subs r0, r0, #1
    mov r3, r3, LSL #1
    str r3, [r1], #+4
    bne doubler
    bx lr
```

My first optimization is that I changed from an incrementing counter to a decrementing one, and used subs instead of a normal sub. The removed a cmp and a conditional branch.

My second optimization is that I removed the stall by moving the subs to be right after the ldrb.

9. (15 points) Arrays

Consider the following definition of median from Wikipedia:

> The median is the value separating the higher half of a data sample, a population, or a probability distribution, from the lower half. In simple terms, it may be thought of as the "middle" value of a sorted data set. For example, in the data set {1, 3, 3, 6, 7, 8, 9}, the median is 6, the fourth number in the sample. The median is a commonly used measure of the properties of a data set in statistics and probability theory.

Write subroutine called `find_median` that returns the median value from an array of integers. Your subroutine should follow the conventions specified by the AAPCS. Your answer should include only the subroutine.

You should *not* assume the array is already sorted. However, you may assume that a subroutine called `sort` exists and that you can use it to sort an array. `sort` takes as parameters the address of the array to sort as well as the number of elements in the array. (Reminder: You are *not* writing `sort`, you just assume it already exists.)

Parameters for `find_median`:

- Address of an array of integers.
- Number of elements in the array. (Guaranteed to be odd.)

Return value of `find_median`:

- The median value from the array.

When writing your answer, correctness should be your first priority. However, among correct solutions efficiency will also be considered when assigning points.

Nothing written on this page will be graded, write your answer on the next page.

Answer for Question 9.

Solution:

There are multiple possible solutions. Here is one.

```
; r4 is the address of the array of integers
; r0 is the number of integers (N)
; r0 will hold the return value
find_median:
    stmfd sp!, {lr}

    ; Call sort.
    ; I need to save r0 first since sort might change it.
    stmfd sp!, {r0}
    bl sort
    ldmfd sp!, {r0}

    ; Calculate the location of the middle element (N/2 + 1)
    mov r0, r0, LSR #1
    add r0, r0, #1

    ; Load that element
    ; The LSL #2 is to multiple the offset by 4 since this is an integer.
    ldr r0, [r4, r0, LSL #2]

    ; Return
    ldmfd sp!, {pc}
```

End of Exam.

Extra Work Page 1.
If you need more space to write your answer to a problem, then write it here and make a note on the problem's main page referring to this space.

Extra Work Page 2.
If you need more space to write your answer to a problem, then write it here and make a note on the problem's main page referring to this space.