CMPE 364 (L51)
Midterm Exam 2
Spring 2017
May 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. 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.**
- **Show your work or you won't get points.**
- **Keep justifications concise and correct.**

Name:

Student ID Number:

Signature:

1. Diagramming Pipeline Stalls

   Consider the following lines of assembly code:

```
add r4, r4, #4
ldr r0, [r4]
sub r3, r3, r0
orr r5, r5, r0
```

   (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	ADD				
2	LDR	ADD			
3	SUB	LDR	ADD		
4	ORR	SUB	LDR	ADD	
5		ORR	SUB	LDR	ADD
6		ORR	SUB		LDR
7			ORR	SUB	
8				ORR	SUB
9					ORR
10					

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

Solution: 9 cycles

2. (10 points) Trial Subtraction

   Compute both the *quotient and remainder* of $3187 \div 11$ using trial subtraction in 10-bits. Remember that you must show all work. In addition, explicitly *label your final answer.*

Solution:

$11 * 2^9 > 3187$

$11 * 2^8 < 3187$ $[3187 - 11 * 2^8 = 371]$

$11 * 2^7 > 371$

$11 * 2^6 > 371$

$11 * 2^5 < 371$ $[371 - 11 * 2^5 = 19]$

$11 * 2^4 > 19$

$11 * 2^3 > 19$

$11 * 2^2 > 19$

$11 * 2^1 > 19$

$11 * 2^0 < 19$ $[19 - 11 * 2^0 = 8]$

Therefore...

$q = 0b0100100001 = 289$

$r = 8$

3. Exceptions Short Answer

(a) (2½ points) Why should an interrupt handler disable interrupts?

Solution: To prevent nested interrupts from occurring. (Where an interrupt handler is interrupted by another interrupt.)

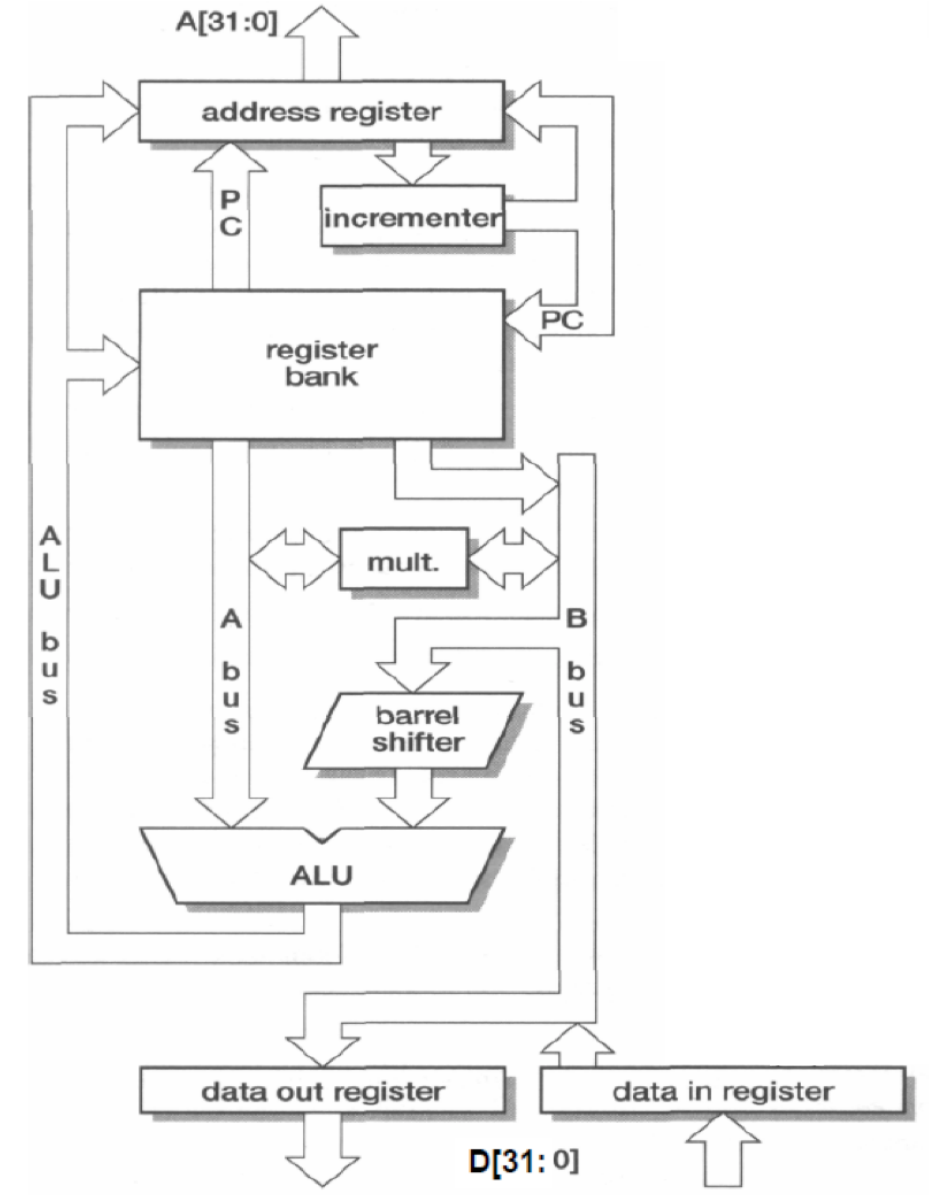
(b) (2½ points) What is an example of a device you would configure with a fast interrupt instead of a regular interrupt? Why?

Solution: There are many valid answers to this question. Here is one.
A flash memory storage device. You want very fast throughput, so the processor needs to receive and handle interrupts very fast.

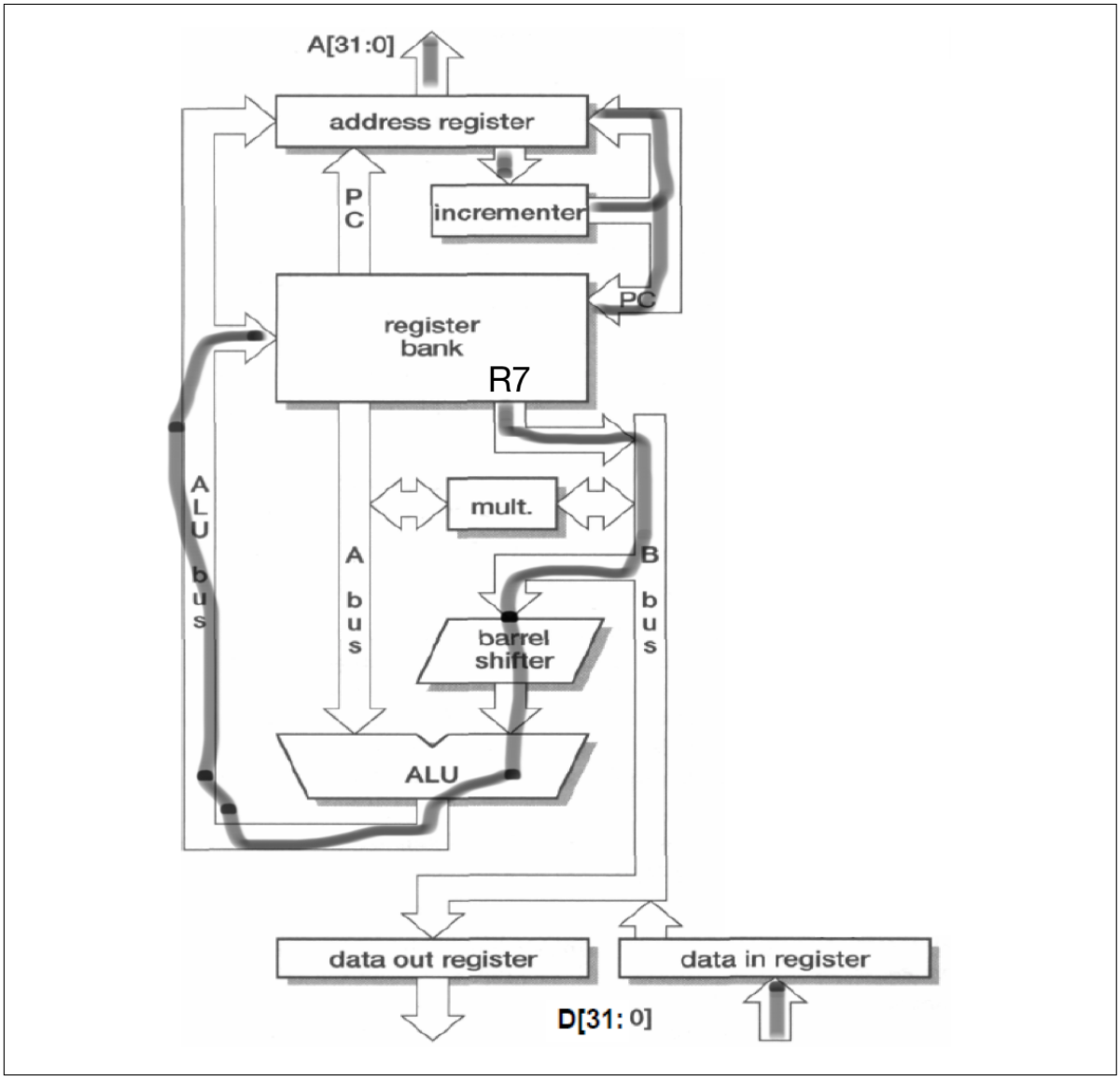
4. (10 points) Datapath

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

```
MOV R8, R7, LSL #8
```



Solution:


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

5. (15 points) Optimization

Consider the following subroutine, `find_max`, which searches an array of unsigned words for the maximum value, and returns that value.

```
; r0 is in the number of entries in the array
; r4 is the address of the array
find_max:
    stmfd sp!, {r4}

    mov r1, #0
    mov r2, #0

s_loop:
    cmp r1, r0
    bge s_done

    ldr r3, [r4], #4

    cmp r2, r3
    bgt skip
    mov r2, r3
skip:
    add r1, r1, #1
    b s_loop

s_done:
    ldmfd sp!, {r4}
    mov r0, r2
    bx lr
```

Optimize this subroutine by reducing the number of cycles it takes to run. (You should *rewrite an optimized version of this subroutine.*) In addition, *write a short paragraph* or bullet point list describing the optimizations you did and how many cycles each one saves over the original. (Note: You should assume the code provided functions correctly. Your optimizations should not alter its functionality.)

You should write your answer on the next page.

Answer space for Question 5. Write your optimized subroutine and your paragraph/list here.

Solution:

```
; r0 is in the number of entries in the array
; r4 is the address of the array
find_max:
    mov r1, r4
    mov r2, #0

s_loop:
    ldr r3, [r1], #4

    cmp r0, #0
    beq s_done

    cmp r2, r3
    movle r2, r3
    sub r0, r0, #1
    b s_loop

s_done:
    mov r0, r2
    bx lr
```

There are three optimizations I did here. (More are possible, too...)

- There is a stall between the `ldr` and the `cmp` that follows it. I moved the load earlier, hence removing this stall and saving one cycle per loop iteration.
- The branch to skip is unnecessary and the same functionality can be handled more efficiently by using a conditional instruction. This saves two cycles per loop when the branch would have been taken, and keeps that same number of cycles otherwise.
- I converted the loop to be a count-down instead of a count-up. This doesn't save any cycles on its own, but it frees up a register. I used that register (`r1`) to store the address of the array, meaning that I no longer modify `r4` and so I can remove the `stm` and `ldm` instructions at the beginning and end of the subrountine. This saves a total of 4 cycles (2 per instruction).

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.