ECE550
PRACTICE Final

This is a full length practice midterm exam. If you want to take it at exam pace, give yourself 175 minutes to take the entire test. Just like the real exam, each question has a point value. There are 115 points from 12 question, so pace yourself accordingly.

Questions:

1. Vocabulary: 10 pts
2. Networking: 10 pts
3. Operating Systems: 10 pts
4. MIPS Assembly: 12 pts
5. Digital Logic: 11 pts
6. Finite State Machines: 10 pts
7. Datapaths: 11 pts
8. Caches: 10 pts
9. Virtual Memory I : 10 pts
10. IO etc: 8 pts
11. Pipelining: 8 pts
12. Short Answer : 5 pts

This is the solution set to the practice exam. The solutions appear in blue.

Question 1 Vocabulary [10 pts]

Match each of the following definitions with the appropriate vocab word:

1. Multiple hard disks combined for performance and/or reliability. M. RAID
2. A memory technology which maintains its state as long as the power is on, but loses its contents when power is turned off. R. SRAM
3. An asynchronous notification of an external event, requiring the attention of the OS. H. Interrupt
4. The structure which holds all of the translations from virtual addresses to physical addresses K. Page Table
5. Discarding incorrect instructions from a pipeline F. Flush
6. A piece of logic which selects between two inputs, based on the value of a third input J. Mux
7. The idea that most of a program's data accesses are likely to be contained within a small range of nearby addresses. Q. Spatial Locality
8. A class of ISAs characterized by simple instructions which are easily implemented in high performance hardware. O. RISC
9. A type of datapath in which the CPI is always 1.0 (by definition). P. Single-cycle
10. The part of the branch predictor responsible for prediciting the taken target of branches (except for returns). B. BTB

A ALU

B Branch Target Buffer

C CISC

D DRAM

E Exception

F Flush

G Hard Disk

H Interrupt

I Multi-cycle

J Mux

K Page Table

L Pipeline

M RAID

N Return Address Stack

O RISC

P Single-cycle

Q Spatial Locality

R SRAM

S Stall

T Temporal Locality

U XOR-gate

Question 2 Networking [10 pts]

This question is multiple-multiple choice: for each question there may be more than one right answer, you need to select **ALL** of the correct answers.

1. Which of the following does IP provide? c,e

   a. Reliable end-to-end data streams.

   b. Ports.

   c. Best-effort deliver.

   d. Encryption

   e. An addressing scheme which allows for scalable routing.

2. Which of the following does TCP provide? a,b

   a. Reliable end-to-end data streams.

   b. Ports.

   c. Best-effort deliver.

   d. Encryption

   e. An addressing scheme which allows for scalable routing.

3. Which of the following is an example of an application layer protocol? a,c

   a http **b** Ethernet **c** ssh **d** TCP **e** IP

4. In TCP, sequence numbers can be used to detect the following problems: b,d,e

   a. Lost ACK when closing a connection.

   b. Out-of-order arrival of data.

   c. Corrupted data.

   d. Duplicate data.

   e. Lost data.

5. The 7-layer OSI model is an example of which of these engineering maxims: d

   a. Fix a performance problem by adding a cache

   b. Fix a functionality problem by adding a level of indirection

   c. Make the common case fast, and the general case possible

   d. Abstraction makes large/complex systems manageable

   e. None of the above.

Question 3 Operating Systems [10 pts]

Fill in the blanks to correctly complete this description of what we learned about operating systems:

The ext2 family of filesystems store meta-data (except for the file's _ name _) in _ inodes _ . This meta-data includes the location of the file's data blocks. This information is stored with 0, 1, 2, or 3 _ levels of indirection _ to make the common case fast and the general case possible. When a file grows, the OS attempts to allocate new blocks in the same block group as the existing blocks of the file to exploit _ spatial locality _ .

A _ process _ is a running instance of a program. To run a new program, an existing program calls _ fork _ to copy itself. The copy then calls exec to overwrite itself with a newly loaded program. The first program, _ init _ is created specially by the OS during the boot code. This process runs forever, and is responsible for reaping orphaned processes—those whose parent _ exited _ before they did.

When the system first boots, the BIOS code reads the boot loader off of the Master Boot Record on the hard disk. It enters the OS in _ 16-bit real _ mode, which is important because the OS must setup its own page tables before _ virtual memory _ can be used.

Question 4 MIPS Assembly [12 pts]

Translate the `arrayManip` function (written in C below) to MIPS assembly:

```
int arrayManip(int * p, int n) {
  int sum = 0;
  int * p2 = p;
  int * pend = &p[n];
  while (p2 != pend) {
      int x = *p2;
      int a = f(x);
      if (a > 4) {
        *p2 = a;
      }
      else {
         sum =  x + sum;
      }
      p2++;
  }
  return sum;
}
```

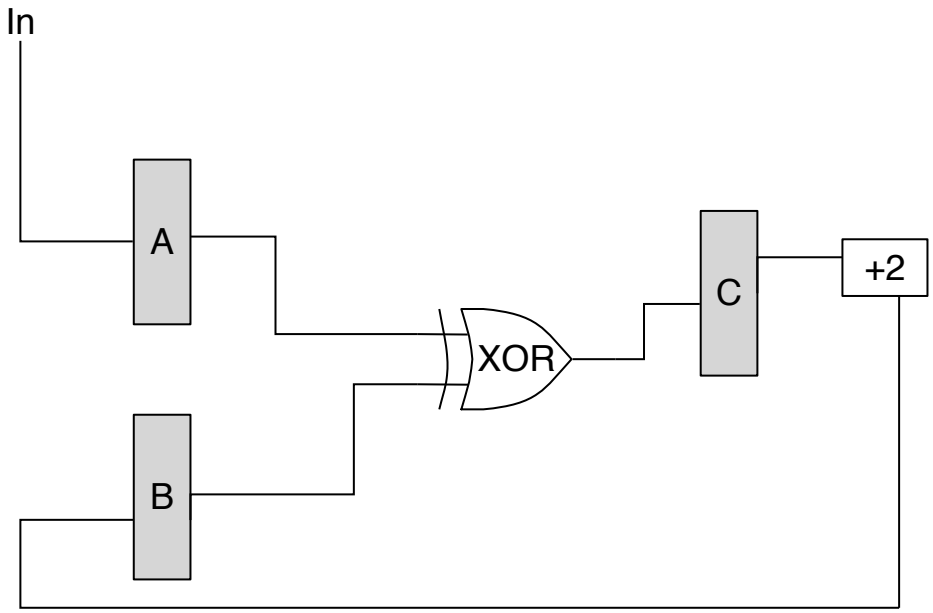
The C code is repeated on the next two page line-by-line. Please do any scratch work you need on this page and answer on the next pages, with your translation of each line of C under that line.

```
# int arrayManip(int * p, int n) {

        addiu $sp, $sp, -32
        sw $fp, 0($sp)
        sw $ra, 4($sp)
        sw $s0, 8($sp)
        sw $s1, 12($sp)
        sw $s2, 16($sp)
        sw $s3, 20($sp)


#  int sum = 0;

        li $s0, 0

#  int * p2 = p;

        move $s1, $a0


#  int * pend = &pn;

        sll $t0, $a1, 2
        add $s2, $a0, $t0


#  while (p2 != pend) {

.L_while:
       beq $s1, $s2, .L_while_end


#      int x = *p2;

        lw $s3, 0($s1)


#      int a = f(x);

        move $a0, $s3
        jal  f
```

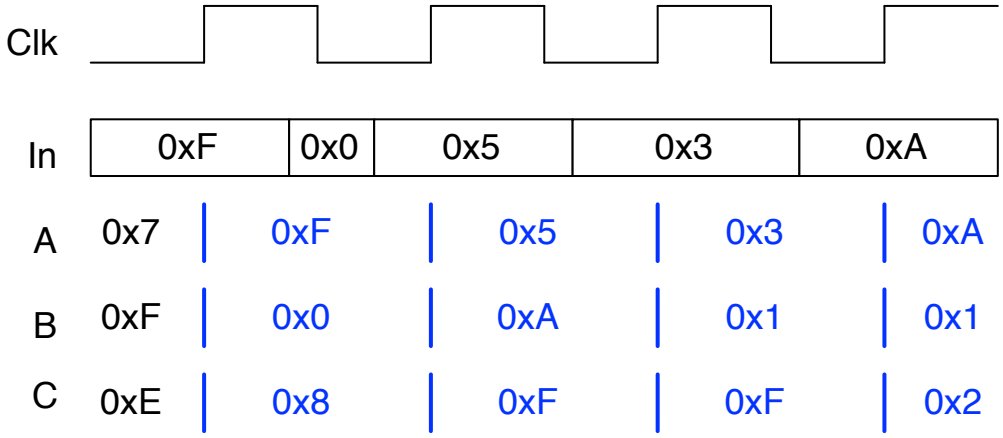
```
#       if (a > 4) {

         li $t0, 4
         ble $v0, $t0, .L_else
#         *p2 = a;

         sw $v0, 0($s1)

#       }
         b .L_endif

#       else {

.L_else:

#           sum =  x + sum;

         add $s0, $s3, $s0

#       }

.L_endif:

#       p2++;

         addiu $s1, $s1, 4

#  }

         b .L_while
.L_endwhile:
#  return sum;
#}
         move $v0, $s3
         lw $fp, 0($sp)
         lw $ra, 4($sp)
         lw $s0, 8($sp)
         lw $s1, 12($sp)
         lw $s2, 16($sp)
         lw $s3, 20($sp)
         addiu $sp, $sp, 32
         jr $ra
```

Question 5 Digital Logic [11 pts]

The following figure shows a digital logic circuit with 3 4-bit registers (A, B, and C). These registers are comprised of DFFs which are triggered by the rising edge of the clock. The circuit has an external input (In) which is also 4 bits.

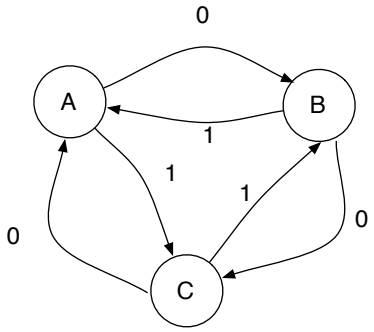


Show the output values of each register over time. Clearly deliinate where the output value changes for each register by drawing a vertical line at the point the value changes.

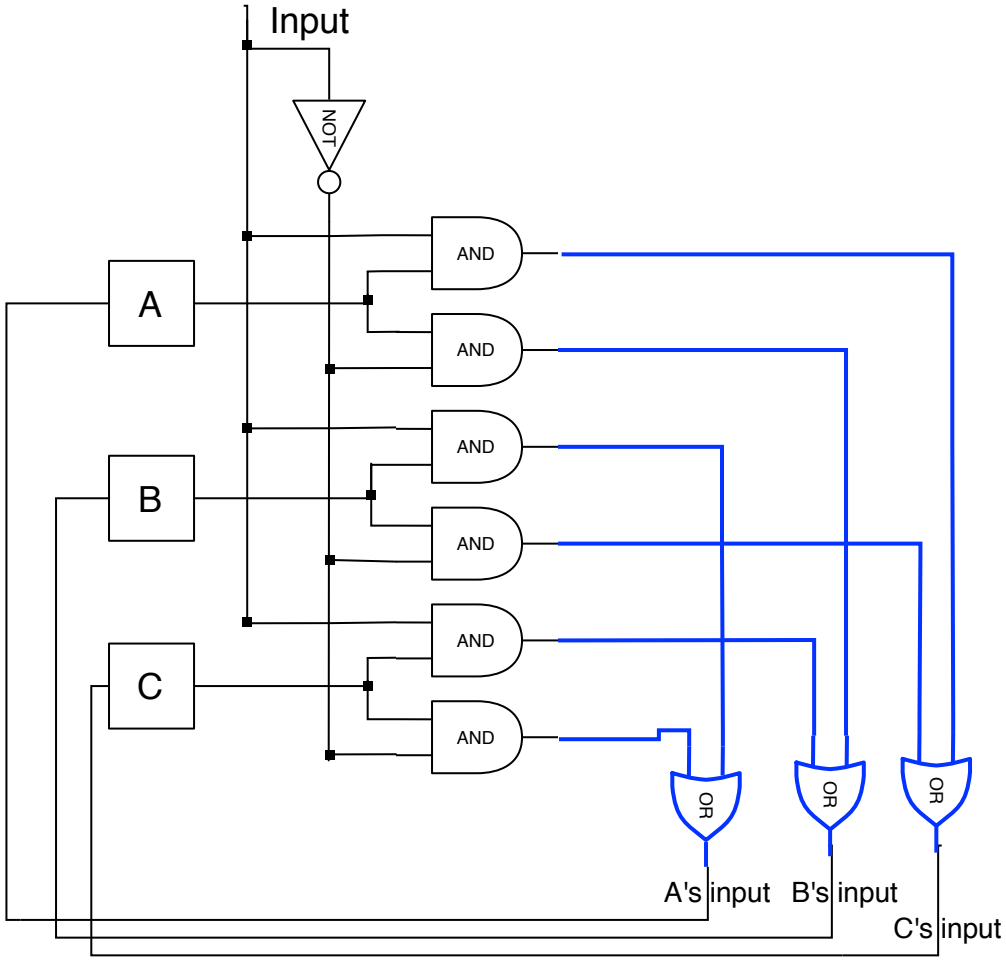


Question 6 Finite State Machines [10 pts]

Consider the following finite state machine diagram (with three states and one input):



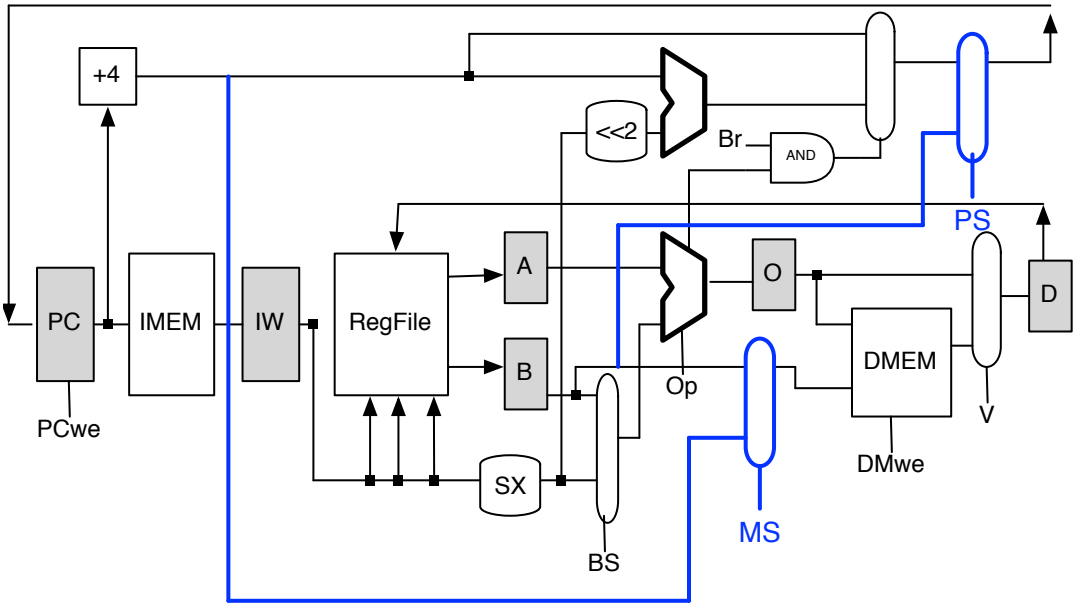
Complete the circuit diagram below which implements the above FSM using a one hot representation and three DFFs (one for each state, labelled accordingly).



Question 7 Datapaths [11 pts]

Modify the basic multi-cycle data-path (shown below) to implement a new instruction, `jrlm $rt, offset($rs)`. This instruction (jump register and link into memory) has the following semantics:

```
Mem[ $rs + offset ]  = PC + 4
PC = $rt
```



After drawing your modifications above, (briefly) explain what happens in the data path on each cycle of the execution of this new instruction (you may not need all the cycles which space has been provided for). The first two are done for you.

1. Instruction is fetched from instruction memory
2. The instruction is decoded and the register file is read.
3. The ALU is used to generate the address (rs + offset)
4. The new MS control signal is set to 1 to pick PC+4 as the data input for data memory. Data memory is write-enabled to perform the store operation. Also, PS is set to 1 to select rt as the next PC, and the PC is write enabled to perform the jump.
5.
6.

Question 8 Caches [10 pts]

A 16-byte cache has 8-byte blocks, has 2 sets, and is 2-way set-associative. The cache initially is empty (all valid bits are off: indicated by a blank box in the table below). The cache receives requests in the sequence listed below. For each address in the sequence (a) split it into the tag, index, and offset; (b) categorize the access as a *hit*, a *compulsory miss*, a *conflict miss*, or a *capacity miss* (You can abbreviate hit=H, Compulsory=O, Conflict=F, Capacity=P); (c) show the new contents of the cache after the access—write the tags for each way, and note which way is LRU. The first one is done for you:

Address	Split Address			Result	Set 0			Set 1		
	Tag	Index	Offset		Way 0	Way 1	LRU Way	Way 0	Way 1	LRU Way
FF	F	1	7	O			0	F		1
F7	F	0	7	O	F		1	F		1
0B	0	1	3	O	F		1	F	0	0
24	2	0	4	O	F	2	0	F	0	0
FB	F	1	3	H	F	2	0	F	0	1
CA	C	1	2	O	F	2	0	F	C	0
08	0	1	0	F	F	2	0	0	C	1
F8	F	1	0	F	F	2	0	0	F	0
35	3	0	5	0	3	2	1	0	F	0
F7	F	0	7	P	3	F	0	0	F	0
32	3	0	2	H	3	F	1	0	F	0

Question 9 Virtual Memory I [10 pts]

Suppose that a system has a 32-bit (4GB) virtual address space. It has 1GB of physical memory, and uses 1MB pages.

1. How many bits are there in the page offset? 20
2. How many virtual pages are there in the address space? 4096
3. How many physical pages are there in the address space? 1024
4. How many bits are there in the virtual page number? 12
5. How many bits are there in the physical page number? 10
6. Some entries of the page table are shown to the right (all values are in hex, and all entries shown are valid). Translate virtual address 0x410423 to a physical address, using the translations in this page table. 0xDD10423

Entry Number	Value
0	1F
1	3C
2	55
3	9C
4	DD
5	EE
6	99
...	...
20	2F
21	4C
22	65
23	AC
24	ED
25	FE
26	100
...	...
40	11F
41	13C
42	155
43	19C
44	1DD
45	1EE
46	199
...	...

Question 10 IO etc [8 pts]

For each question, circle the best answer. If none of the selections are appropriate, then choose "e. None of the above"

1. Which of the following is an advantage of interrupts over polling? b

   a. Interrupts are a simpler mechanism for both the hardware and software.

   b. Interrupts allow for more efficient CPU utilization.

   c. Interrupts are compatible with virtual memory, while polling is not.

   d. Polling causes bad hit rates in the L1 instruction cache.

   e. None of the above

2. What three operations (in the correct order) are required to read data from the hard disk? d

   a. jal, sw, jr

   b. Wait for rotation, seek, read data as it spins under

   c. Interrupt, seek, read data as it spins under

   d. Seek, wait for rotation, read data as it spins under

   e. None of the above

3. In order to acheive high performance from a hard disk, software must c

   a. Track the rotational position of the hard disk to estimate rotational delay.

   b. Use virtual memory.

   c. Request large sequential reads/writes.

   d. Directly access the disk without the overhead of the OS.

   e. None of the above

4. Which of the following would **not** cause an exception? b

   a. Division by zero.

   b. A branch is mis-predicted.

   c. Attempting to execute a priveledged instruction in a normal application.

   d. A load or store does not have a valid virtual to physical translation.

   e. None of the above (all cause exceptions).

Question 11 Pipelining [8 pts]

Fill in the blanks below to correctly complete the explaination of some of the difficulties and solutions of pipelining.

One complexity that can arrise with pipeling is _ data _ hazzards—in which the producer has not yet written its result to register file before the consumer reads it. The original MIPS ISA fixed this with software interlocks. However, most ISAs avoid these for both _ compatability _ and performance reasons. Hardware interlocks fix the first problem, but still suffer poor performance. A better solution is _ bypassing _ , which is implemented by adding muxes to the X and M stages to choose between values coming from older in-flight instruction or the register file. Even when all combinations of (word in previous blank) are implemented, there is one situation where the pipeline must stall: a _ load _ instruction followed immediately by a dependent use.

Another type of hazzard is _ control _ hazzards, which arise from branches. These hazzards are best addressed through speculation—guessing the correct outcome of the branch before it executes. In doing so, the pipeline may guess incorrectly, and need to _ flush _ incorrect instructions.

Question 12 Short Answer [5 pts]

1. Compare and contrast caches and virtual memory. Give at least one similarity and one difference between the two. For the difference, explain *why* this difference exists.

Answer:
(Many possible)
Similarity: both split memory into fixed sized chunks (blocks/pages), and manipulate memory at this granularity.
Difference: In virtual memory, software (the OS) makes a replacement decision, while in caches, the hardware makes the replacement decision. This difference exists because Tmiss is so much larger for memory (which misses to disk) than any other level of the memory hierarchy—this justifies the extra time for software transfer control into a software routine to make a complex decision, in the hopes of reducing %miss.