EE380 Spring 2018 Exam 0

Your name is: `Sample solution`

Your account name (as used for assignments) is:

The two questions you skipped are:

There are 15 questions. Each of the first 14 questions is worth 8 points; **you must answer any 12 and indicate above which 2 you skipped** (no extra credit). **You also must answer question 15**, which is worth just 4 points. For each question with boxes, ☐, in front of each potential answer, read the question carefully to see if you should check a single answer or all answers that apply. The short answer questions should get **short** answers; excessively long answers may be considered incorrect.

The last page is a reference sheet; feel free to tear it off. You do not need to turn it in with your exam. **No other reference materials of any kind are permitted, nor are any electronics (e.g., no calculators nor cell phones).**

1. *For this question, mark all answers that apply.* Which of the following attributes is/are typical of a *high-end* PC built in 2018?

- [x] Main memory is usually between 2GB and 16GB
- [x] The processor clock period is a few hundred picoseconds
- [x] There is a graphics processing unit that does parallel computation
- [x] There are at least 2, and probably more, cores in your processor chip

2. *For this question, mark all answers that apply.* In computers, most aspects of system performance are improving exponentially over time. Which of the following good things are happening over time?

- [ ] Processor designs are getting simpler
- [x] The capacity of main memory is increasing
- [x] The power consumed per transistor in a processor is decreasing
- [x] The amount of work a computer can do in one second increases faster than the speed at which transistors can switch

3. Which type of memory, SRAM or DRAM, would you expect to be **most vulnerable** to electronic/electrical noise? Why?

DRAM would be more vulnerable because it uses a tiny charge on a capacitor to store a bit; SRAM uses D flip-flops. DRAM also uses an analog sense amplifier.

4. A 3D printer contains a computer that must send "step" signals to each of the X, Y, and Z stepper motors very quickly. Suppose that the basic motion control code requires executing 10 loads, 10 adds, 5 jumps, and 5 stores. The processor takes 10 clock cycles per load, 5 per add, 20 per jump, and 10 per store. How many clock cycles does this code take to execute? How many microseconds does this code take to execute using a 10MHz clock?

```
load    10    10
add     10     5
jump     5    20
store    5    10

(10*10)+(10*5)+(5*20)+(5*10) =
100+50+100+50 = 300 clock cycles

10MHz is 100ns clock period
300 * 100ns = 30,000ns = 30us
```

5. *For this question, mark all answers that apply.* A competing 3D printer was just announced that prints faster than yours; their motion control code sends step commands every 10 microseconds. Your stepper motors are capable of stepping that fast... which of the following modifications could allow your control logic to at least match that speed?

■ Replace the processor with a more expensive one that can use a 10ns clock period

☐ Replace the jump instructions with branch instructions that execute in 5 clock cycles each

☐ Re-write the code so all the values are kept in registers, eliminating all the loads and stores

■ Implement your own pipelined processor in an FPGA; it will run with a 5MHz clock, but will complete one instruction every clock cycle

6. *For this question, mark all answers that apply.* Which of the following statements about ISAs is **true**?

■ ISA stands for Instruction Set Architecture

☐ The ISA specifies the minimum clock frequency

☐ The ISA specifies if there is a divide hardware module

■ The ISA specifies the exact bit pattern that is to be used to encode each instruction

7. Consider the implementation architecture discussed in class and summarized at the end of this exam. Assume that the propagation delay associated with `MARin` is 1ns, `Yin` is 2ns, `PCin` is 4ns, `REGout` is 8ns, and `SELrd` is 16ns. Given that the following state is the limiting factor on clock speed, what is the shortest allowable clock period? Why?

SELrd, MARin, Yin, PCin, REGout

SELrd -> REGout -> MARin 16+8+1 = 25ns
SELrd -> REGout -> Yin 16+8+2 = 26ns
SELrd -> REGout -> PCin 16+8+4 = 28ns

Thus, 28ns is the fastest period that will latch correct results on all active circuit paths

8. Using the implementation architecture discussed in class and summarized at the end of this exam, write an RTL control sequence that would implement an increment instruction, **inc rd**, so that **rd=(rd+1)**. Add your control sequence to the following:

```
Start:
  PCout, MARin, MEMread, Yin
  CONST(4), ALUadd, Zin, UNTILmfc
  MDRout, IRin
  Zout, PCin, JUMPonop

Inc:
```

SELrd, REGout, Yin
CONST(1), ALUadd, Zin
Zout, SELrd, REGin, JUMP(Start)

9. *For this question, mark all answers that apply.* Consider the implementation architecture discussed in class. The following three RTL control sequences all implement functionally identical instructions. You should assume that the control logic will repeat any state containing **`UNTILmfc`** until **2 clock cycles after** the clock cycle in which **`MEMread`** is asserted (i.e., memory takes two clock cycles to respond). Which of the following statements is/are correct?

```
Start:
  PCout, MARin, MEMread, Yin
  CONST(4), ALUadd, Zin, UNTILmfc
  MDRout, IRin
  Zout, PCin, JUMPonop
  Halt

InstA:
  SELrt, REGout, MARin, MEMread
  SELrs, REGout, Yin
  SELrd, REGout, ALUadd, Zin
  Zout, Yin, UNTILmfc
  MDRout, ALUand, Zin
  Zout, MDRin, MEMwrite
  Zout, SELrd, REGin, JUMP(Start)

InstB:
  SELrs, REGout, Yin
  SELrt, REGout, MARin, MEMread
  SELrd, REGout, ALUadd, Zin, UNTILmfc
  Zout, Yin
  MDRout, ALUand, Zin
  Zout, MDRin, MEMwrite, SELrd, REGin, JUMP(Start)

InstC:
  SELrs, REGout, Yin
  SELrd, REGout, ALUadd, Zin
  Zout, Yin
  SELrt, REGout, MARin, MEMread
  UNTILmfc
  MDRout, ALUand, Zin
  Zout, MDRin, MEMwrite, SELrd, REGin, JUMP(Start)
```

5 cycles common to all...

7 cycles (memory delay hidden)

7 cycles (2 cycles for UNTILmfc)

8 cycles (2 cycles for UNTILmfc)

☐ InstA would take more clock cycles than InstB

☐ InstB would take more clock cycles than InstC

☐ InstA would take more clock cycles than InstC

☐ InstA, InstB, and InstC all take the same number of clock cycles

10. Using the implementation architecture discussed in class and summarized at the end of this exam, write an RTL control sequence that would implement the increment memory instruction, **`incm immed(rs)`**, so that **`memory[rs+immed]=memory[rs+immed]+1`**. Note that you don't have to compute **`rs+immed`** twice. Add your control sequence to the following:

```
Start:
  PCout, MARin, MEMread, Yin
  CONST(4), ALUadd, Zin, UNTILmfc
  MDRout, IRin
  Zout, PCin, JUMPonop

Incm:
```

```
SELrs, REGout, Yin
IRimmedout, ALUadd, Zin
Zout, MARin, MEMread
CONST(1), Yin, UNTILmfc
MDRout, ALUadd, Zin
Zout, MDRin, MEMwrite, JUMP(Start)
```

11. Using the implementation architecture discussed in class and summarized at the end of this exam, write an RTL control sequence that would implement the shift-and-add instruction, **`saa rd,rs,rt`**, so that **`rd = (rs + (rt * 2))`**. Add your control sequence to the following:

```
Start:
  PCout, MARin, MEMread, Yin
  CONST(4), ALUadd, Zin, UNTILmfc
  MDRout, IRin
  Zout, PCin, JUMPonop

Saa:
```

```
    SELrt, REGout, Yin
    Yout, ALUadd, Zin
    SELrs, REGout, Yin
    Zout, ALUadd, Zin
    Zout, SELrd, REGin, JUMP(Start)
```

12. *For this question, mark all answers that apply.* Which of the following statements about performance metrics is *true*?

- ■ IPC is 1/CPI
- ■ In a multi-core UNIX system, the total **user time** can exceed the elapsed **real time**
- □ Throughput is greater executing a few long-running jobs than executing lots of short jobs
- ■ Many processors now have tick counter registers that can be read to find out how many clock cycles have elapsed

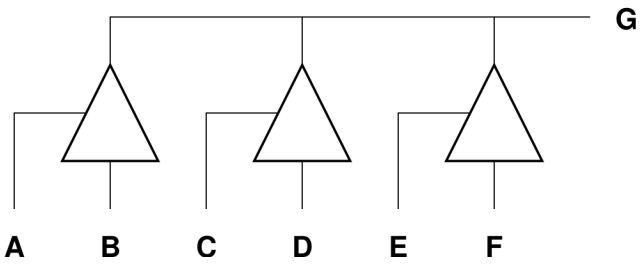
13. There is now lots of support for using your cell phone as a head-mounted "3D" stereo display. It gets strapped horizontally in front of your eyes, and the left and right images are drawn on the portions of the display that are seen by each eye. Suppose you have created an algorithm that can synthesize left and right images from an ordinary HD video stream within your phone, but not at the required 30 frames per second (FPS) — your algorithm can only process 10 FPS. Nearly all the execution time for your code is in two functions: **`analysis()`** and **`render()`**, which respectively take 60% and 40% of the total execution time. There doesn't seem to be anything you can do about the speed of the **`render()`** function. However, your phone has an internet connection, so the **`analysis()`** function might be sped-up a lot by running it on "the cloud" (ok, really a supercomputer server) instead of inside your phone. According to **Amdahl's law**, what is the *maximum speed-up* you might get by speeding-up only the **`analysis()`**? Could that be sufficient for the system to meet its 30 FPS goal?

`analysis() is 60%`
`render() is 40%`

`Speedup if analysis() goes to 0 is (60+40)/40 = 2.5X`

`It was 10FPS, needs 30FPS... but can't get better than 25FPS by changing analysis() alone: NO, goal isn't met.`

14. *For this question, mark all answers that apply.* Consider the following circuit. Which of the following sets of values for A, B, C, D, E, and F would result in G being driven to a stable output of 1?



☐ A=1, B=1, C=1, D=1, E=1, and F=0

■ A=1, B=1, C=0, D=1, E=0, and F=0

☐ A=1, B=0, C=0, D=0, E=0, and F=1

■ A=0, B=1, C=0, D=1, E=1, and F=1

15. Suppose that you are writing an application that will be continuously logging lots of new data (updating and extending files) in a computer placed in remote location. You could log the data to a hard disk drive or an SSD (solid state disk). Which would you use? Either choice can be valid, but you must give a valid technical reason why your choice is good.

SSD: no moving parts (more reliable in remote location), potentially lower power, faster.

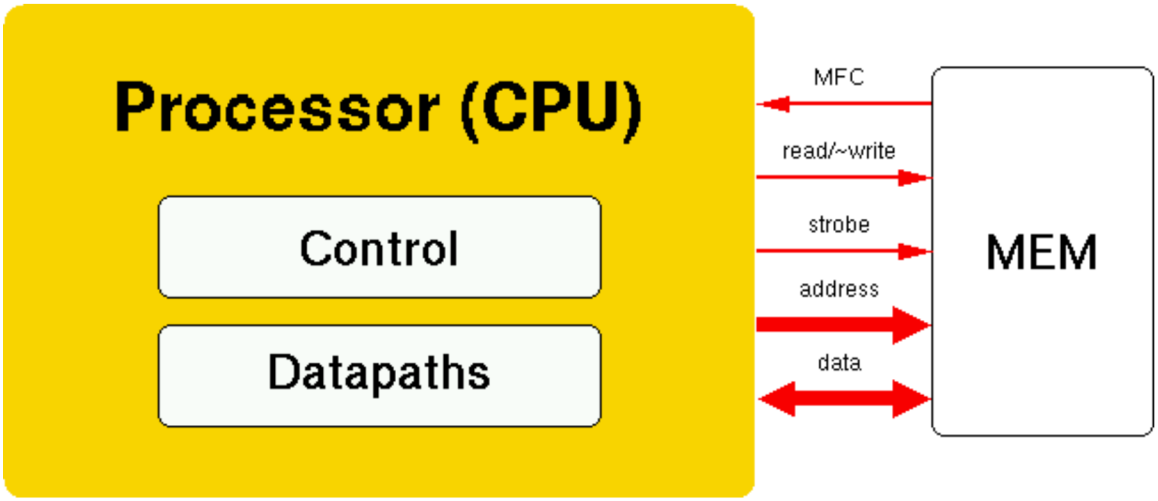
Hard Disk: can handle lots of writes without wearing out, higher capacity.

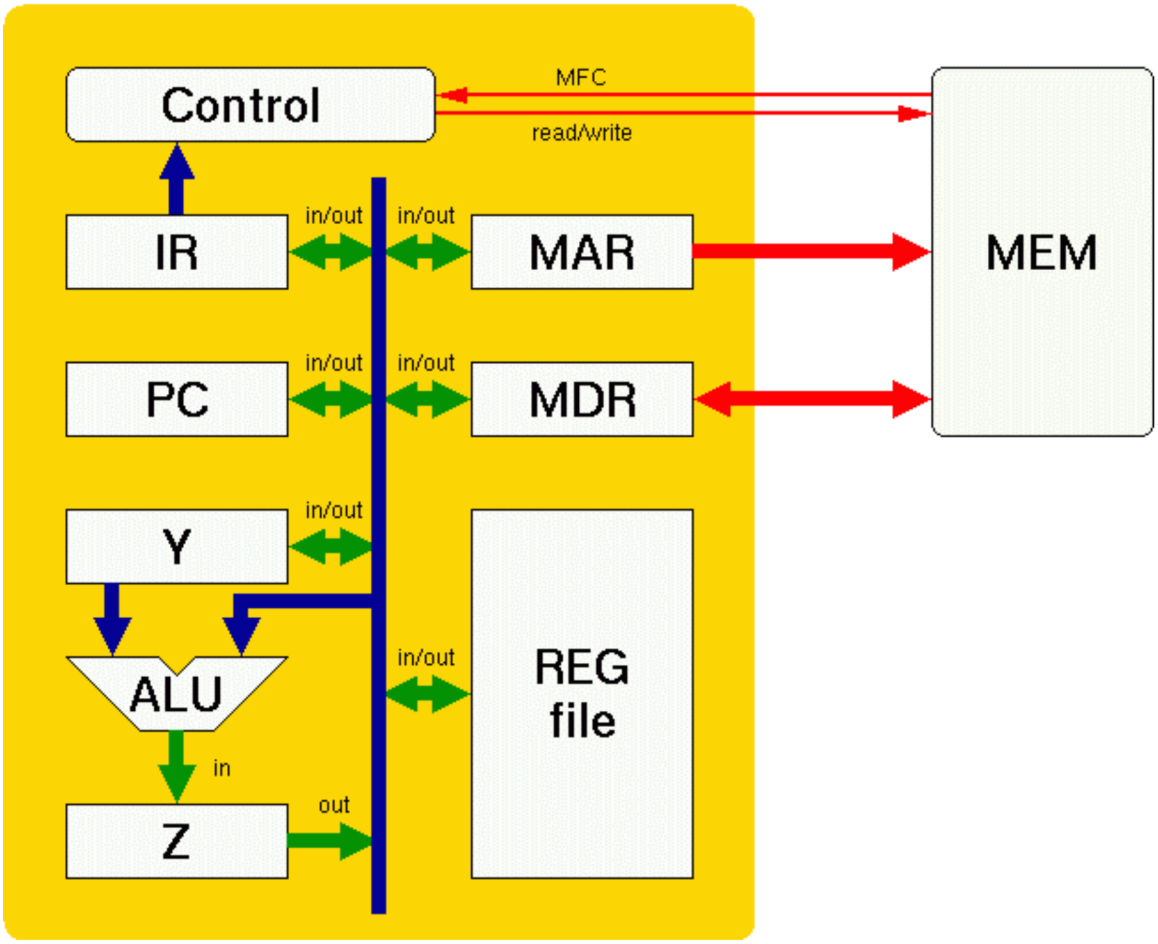
There are many ways to justify either choice....

This page was intentionally blank...

until I printed this text on it.

Reference for Simple Processor Implementation





Control Signal	Effect
ALUadd	Configures the ALU to add its inputs
ALUand	Configures the ALU to bitwise AND its inputs
ALUxor	Configures the ALU to bitwise eXclusive OR its inputs
ALUor	Configures the ALU to bitwise OR its inputs
ALUsll	Configures the ALU to shift left logical; the result is (bus << Y)
ALUslt	Configures the ALU to compare its inputs; the result is (Y < bus)
ALUsrl	Configures the ALU to shift right logical; the result is (bus >> Y)
ALUsub	Configures the ALU to subtract the bus input from Y
CONST(value)	Places the constant *value* onto the bus
HALT	Halt the machine (stop the simulator without error) at the end of the current state
IRaddrout	Tri-state enables the portion of the Instruction Register that contains the (26 bit, MIPS "J" format) address, along with the top 6 bits of the Program Counter, to be driven onto the bus
IRimmedout	Tri-state enables the portion of the Instruction Register that contains the (16 bit, MIPS "I" format) 2's complement immediate value to be sign-extended to 32 bits and driven onto the bus
IRin	Latches the bus data into the Instruction Register at the trailing edge of the clock cycle
IRoffsetout	Tri-state enables the Instruction Register's shifted and sign extended value from the offset field to be driven onto the bus (used for branches)
JUMP(label)	Microcode jump to *label*
JUMPonop	Microcode jump to label named like the opcode; e.g., if an "Addi" is in the IR, jumps to the microcode label Addi
MARin	Latches the bus data into the Memory Address Register at the trailing edge of the clock cycle
MARout	Tri-state enables the Memory Address Register's output to be driven onto the bus
MDRin	Latches the bus data into the Memory Data Register at the trailing edge of the clock cycle
MDRout	Tri-state enables the Memory Data Register's output to be driven onto the bus
MEMread	Initiate a memory read from the address in the MAR; it will take some number of clock cycles for memory to respond with MFC (memory fetch complete)
MEMwrite	Initiate a memory write using the data in the MDR and the address in the MAR; in this simple design, you may assume that a memory write takes precisely 1 clock cycle
PCin	Latches the bus data into the Program Counter at the trailing edge of the clock cycle
PCinif0	Only if the value in Z is zero, latch the bus data into the Program Counter at the trailing edge of the clock cycle
PCout	Tri-state enables the Program Counter's output to be driven onto the bus
REGin	Latches the bus data into whichever register is selected by SELrs, SELrt, or SELrd; the value is latched at the trailing edge of the clock cycle
REGout	Tri-state enables the output of whichever register is selected by SELrs, SELrt, or SELrd; the selected value is driven onto the bus
SELrs	Selects the rs field of the IR to be used to control the register file's decoder
SELrt	Selects the rt field of the IR to be used to control the register file's decoder
SELrd	Selects the rd field of the IR to be used to control the register file's decoder
UNTILmfc	Repeat this state until the memory has issued MFC (memory fetch complete), indicating that the fetched value will be valid to read from the MDR in the next clock cycle
Yin	Latches the bus data into the Y register at the trailing edge of the clock cycle; this register is needed because, with only one bus, one of the two operands for a binary operation (e.g., Add) must come from somewhere other than the bus
Yout	Tri-state enables the Y register's output to be driven onto the bus
Zin	The ALU is always producing a result, but we only make note of that result if we latch the ALU's output into the Z register at the trailing edge of the clock cycle
Zout	Tri-state enables the Z Register's output to be driven onto the bus