CIS 350 – INFRASTRUCTURE TECHNOLOGIES
HOMEWORK #4

Student Name(s):__Cedric Fenn____
(You may do this homework in groups of 2 students maximum.)

Topics: The CPU and Memory (Chapters 7 and 8)

Ex. Show an instruction format that could be used to move data or perform arithmetic between two registers (the source register and the destination register). Assume that the instruction is 32 bits wide and that the computer has 8 general-purpose registers. If the op code uses 7 bits, how many bits are spares, available for other purposes, such as special addressing techniques? Draw a detailed diagram with the instruction format. Figure 7.21 on p. 228 in your textbook depicting a "register to register" format may be very helpful.

22

Ex. 7.3, p. 232

2^{48} = 281,474,976,710,656 bytes of memory

Ex. 7.7 a (only), p. 232

Shifting 2 bits to the left result in the unsigned number multiplied by 4. The shift of 1 bit to the right will divide the number by 2.

Ex. Suppose that the following instructions are found at memory locations 25 and 26. Suppose that the following data are found at memory 55 and 56.

Address	Instruction	
25	LDA 55	
26	ADD 56	Addresses 25-26 represent the program area
	Data	
55	125	Addresses 55-56 represent the data area
56	10	

Show the contents of the PC, the MAR, the MDR, the IR, and the A as each step of the fetch-execute cycle is performed for instructions at addresses 25 and 26. (The machine cycle for instruction LDA I worked in class on the white board would be helpful. See page 5 in the lecture notes for chapter 7. I asked students to take notes. Also, look at Assignment One in in-class small group activity #4.)

Instruction: 25 LDA 55

	PC	MAR	MDR	IR	A
1. PC→ MAR	25	25	LDA 55	?	?
2. MDR→ IR	**25**	**25**	**LDA 55**	**LDA 55**	**?**
3. IR[addr] → MAR	**25**	**55**	**125**	**LDA 55**	**?**
4. MDR→ A	**25**	**55**	**125**	**LDA 55**	**125**
5. PC+1→ PC	**26**	**55**	**125**	**LDA 55**	**125**

Instruction: 26 ADD 56

	PC	MAR	MDR	IR	A
1. PC→ MAR	**26**	**26**	**ADD 56**	**LDA 50**	**125**
2. MDR→ IR	**26**	**26**	**ADD 56**	**ADD 56**	**125**
3. IR[addr] → MAR	**26**	**56**	**10**	**ADD 56**	**125**
4. MDR+A→ A	**26**	**56**	**10**	**ADD 56**	**135**
5. PC+1→ PC	**27**	**56**	**10**	**ADD 56**	**135**

Short essay questions. Your answers should capture the essence of the questions. There is no credit for 1- or 2-sentence answers.

Ex. 8.4 and 8.5, p. 263.

8.4
Pipelining reduces the average number of steps in the execution of the fetch-execute cycle by executing multiple instructions at the same time. This can increase the CPU's output. Pipelining can fetch the first instruction then fetch the second instruction while the first is still being executed. If one instruction is being executed another instruction can be fetched simultaneously if it not dependent on the prior instruction to be finish executing first.

8.5
Branch instructions can reduce performance by invalidating all the instructions in the pipeline. One method would be to pre-fetch the target instruction before it is needed, called a branch prediction. The prediction is based on the history of execution of the same instruction. Another method to overcome this problem would be having separate pipelines for both possibilities. Have one condition where the branch instruction is taken and another condition that does not take the branch instruction, cancelling the invalid branch.

Ex. 8.16, p. 263. Also briefly explain how cache memory works.

When a system has multiple levels of cache memory, L2 will always have more memory than L1 because the two cache levels cannot contain the same data so L2 will have more cache to improve performance. This increases the likelihood that requests to the secondary cache can be met without going out to the main memory every time. Cache memory is implemented in SRAM chips on the CIP chip. It has high speed memory of sizes between 512 KB to 3 MB, increasing in size the higher the level. Cache is organized in blocks/lines. Each block has a tage that identifies the location in main memory that relates to the data in that block. When cache memory is full, a block is replaced.

Ex. 8.17, p. 263

Multicore computers have multicore processors meaning multiple CPUs operating on a single integrated circuit. Each CPU in the processor is a core. This will increase the processing power and allow the computer to multitask. A four-core processor would have four times the processing power of the single-core processor.

Ex. 8.18, p. 263

Master-slave multiprocessing or symmetrical multiprocessing are two ways to configure multiprocessing systems. Master-slave multiprocessing manages the system, controls all resources and scheduling, and assigns tasks to slave CPUs. Symmetrical multiprocessing gives equal access to resources for each CPU and determines what to run using a standard algorithm. Symmetrical multiprocessing is more effective for general purpose computing. Master-slave multiprocessing is more computationally intense and more effective in specialized processing tasks.