1. (15 points) Given 8-bit words
   a) (5 points) What is the binary representation of $9D$?
   b) (5 points) If $B4$ is in 2's complement form, then what is its decimal value?
   c) (5 points) If $E7$ is in signed-magnitude form, then what is its decimal value?

2. (30 points) Convert the following assembly language mnemonics to machine code.
   a) (15 points) For an 8x86 in Intel form, add cl, [si + bp + 729h]. The opcode for this add is $02$. To facilitate partial credit, I suggest that you provide the mode, reg, r/m, and displacement separately before combining them.
   b) (15 points) For a MIPS, OR $t7$, $a2$, $s3$. OR has opcode 0, and func 0x25. To facilitate partial credit, I suggest that you provide the encoding for each part of the instruction separately before combining them.

3. (25 points) Using the instruction times listed in Appendix A.10, p. 367, fill in the machine cycles for each instruction and then write the timing expression for the following code in term of N. N’s address is stored in the ! $5

   Cycles
   BGN # 2
   LDX# 0
   LOOP: LDA ! $4
   ADA# 1
   OUTW+ ! 6
   AOC* ! $5 ; N
   JLT LOOP
   FIN # 2
   RTN

   Timing expression: ___________________________

4. (82 points) Write an Intel register neutral function for a min function that will set the int pointed to by minValue to the minimum value found in the array. Please use the 32 bit registers, e.g. eax, not ax. Comments are unnecessary but would be helpful. The signature of min() is void min(int *array, int arrayCount, int *minValue);
   You should assume the following declarations:

   array dd 1
dd 5
dd -9
dd 18
arrayCount dd 4
minValue dd 0

   a) (17 points) Provide the Intel assembly code you would write to place the parameters on the stack, call min(), and return the stack to its proper state after calling min(). You must use arrayCount, and not 4.
   b) (65 points) Provide all the Intel assembly code for the min() function.
5. (50 points) CUSP function
   a) (39 points) Write a "register-neutral" CUSP subroutine for `strcmp()`. The C declaration is:
   ```c
   int strcmp(const char *str1, const char *str2);
   ```
   This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating '\0' character is reached. It returns an integral value indicating the relationship between the strings: A zero value indicates that both strings are equal. A value greater than zero indicates that the first character that does not match has a greater value in `str1` than in `str2`; And a value less than zero indicates the opposite. Comments are unnecessary, but would be helpful.

   b) (11 points) Given the following, write the CUSP code to call the subroutine `strcmp`, and correct the stack pointer.
   ```
   s1: .char 'Hello'
   .word 0          ; terminate with '\0'
   s2: .char 'Hi'
   .word 0          ; terminate with '\0'
   ```

6. (30 points) CUSP is definitely a CISC CPU. Suggest 3 changes to the instruction set and addressing modes that would bring it more in line with a RISC CPU. Assume we still have seven registers, ACC, PC, XR, SP, FP, flags, and IR, but they may be re-wired.

7. (25 points) In a few sentences, describe how compilers and operating systems facilitate machine independent programming.

8. (25 points) Operating systems
   a) (15 points) In a sentence or two explain how modern preemptive operating systems regulate application access to the CPU?

   b) (10 points) Before preemptive operating systems, there was cooperative multitasking in which applications voluntarily surrendered the CPU to the operating system. In one sentence, explain how an application could surrender control of the CPU to the operating system.

9. (20 points) CPU Design
   The Intel 8086 describe in the text could only access one megabyte of RAM. Now they can access more. Suggest the changes you make so that a later version of x86 CPUs could access four gigabytes of RAM. Beware, this requires multiple changes in the CPU design.