

Microprocessor Course

Homework#1 - Assembly Programming for 8086

1. The gray encoding method is a standard process for mapping n -bit binary vectors to n -bit gray-encoded values. The main advantage of the gray-coded values is that in each transition only one bit will be changed. This issue makes the gray-encoding process a suitable alternative for encoding state machines. It is also widely used in communications. The encoding process of 3-bit binary values to 3-bit gray-code vectors is shown in the following table:

Binary Value	Gray Value
000	000
001	001
010	011
011	010
100	110
101	100
110	101
111	111

You are required to write an assembly code to cover the addition process for two 16-bit gray-vectors, by reading them from internal RAM and store the gray-coded output result in another address in the internal RAM. Try to achieve the fastest execution time and minimum memory usage in your assembly code. You are allowed to use only the 16-bit general purpose registers in 8086.

2. Interfacing peripheral devices can be handled by 8086. An important issue in those data transfers is the error checking process. Suppose we have a 16-bit bus for communicating a peripheral device with 8086. The physical interface can operate up-to 4MHz of frequency, which means that each new 16-bit vector can become available in 250ns. Consider the last byte of data within a transfer is determined by "0000001P", where P is the parity bit of the previous bytes (the last byte is not taken into consideration). Also suppose that there is no limit for the length of data (number of bytes). Write an assembly code to check the parity error of the received data, while storing the data in the internal RAM in a pipeline fashion. The starting memory address

for the storing process is "00H". Try optimizing the execution time and consume as lower number of general purpose registers as you can. Moreover, what is the maximum rate of data transfer in your system? In order to achieve higher frequencies what do you propose? (For easy calculations suppose the maximum achievable frequency in executing each instruction in 8086 is 10MHz).

3. Although floating-point arithmetic calculations for unsigned fractional numbers is much more accurate compared to the fixed-point calculations, the fixed-point approach is more suitable for software programming due to its simplicity. We would like to write an assembly code for multiplying two unsigned fixed-point numbers with 32-bits length for their integer part. Choose the suitable bit length for the fractional part and write an assembly code to achieve the following constraints, respectively:

- Multiplication execution time must be lower than 4 μ s. (f_{max} in 8086 is 10MHz for all the instructions)
- Achieve the highest accuracy of multiplication by choosing the highest possible bit-length for the fractional part.
- The third optimization priority is minimizing the memory usage. Try benefiting from the 32-bit *eax* and *edx* registers as more as possible. In addition to these two registers you are not allowed to use more than 6 numbers of double-word registers.

Provide your whole source code in a sub-routine structure. Then write an assembly code to read two unsigned fixed point numbers and multiply them by calling your sub-routine structure.

Grades

Problem#1: 20%

Problem#2: 35%

Problem#3: 45%

You can use any 8086 simulators to verify your assembly codes. Provide a complete report and email it to sarbishei@ee.sharif.edu by 9:00am on Saturday, 24th of Farvardin. Your reports must fulfill the following issues:

- *High-level pseudo-codes of your programs.*
- *Low-level assembly source codes with their comments.*
- *8086 simulator results after verifying your assembly codes.*

*Regards,
O. Sarbishei*