CMPE 230 Systems Programming

Project 1

2017400069 - 2017400162

In this project, we implemented an interpreter for an assembly language of a hypothetical 8086- like CPU called HYP86, in C++ language. We used 32 methods including main and some global variables in order to reach from all methods.

For the register structure, we used 7 global unsigned short and in order to reach them we used a map called registerMap. This map is used for writing value in register and read it.

For the subregister structure, we used 8 global pointer and in order to reach them we used a map called subRegisterMap. This map is used for writing value in subRegister and read it.

For the memory structure, we used an unsigned char array. This array can be called in each method when is necessary.

For holding instructions line by line, we used a global vector named instructions. This vector used only in main for seperating lines to words. Those words are kept in a 2D string array named parsedInst[][].

For the variable structure, we used a map called variables. This map holds the type of variable and memory index of this variable.

For control mechanism, we used a global boolean called control. This bool can be changed in methods. İf it is false we can assume that there are an error.

For flag structure, we used 5 global boolean. These bools can be changed in methods.

**Int Main:**

This method read instructions from file and kept them in instructions vector. When it reaches “int 20h” it waits for variables and kept them in **variables** . After reading part, it starts control strings in parsedInst array, which is loaded with string tokenizer. In this section, code controls hexadecimals and decimal numbers, change to decimal if necessary. Then if statements control all possible commands.

**Void Registers** **():**

This method updates values of registers in order to values in registerMap.

**String Trim** **(string str):**

This method trims blank spaces in instructions.

**Int variableValue** **(string var):**

This method takes variable name in string format and calculate its value from memory.

**Int hexadecimalToDecimal** **(string hex):**

This method convert hexadecimal to decimal.

**Void auxiliary** **(int n, int m):**

This method changes auxiliary flag when its necessary.

**Void sign** **(int a, int b):**

This method changes sign flag when its necessary.

**Void carry** **(int a,int b,int bb,int d):**

This method changes carry flag when its necessary.

**Void overflow** **(int a,int b,int bb,int c):**

This method changes overflow flag when its necessary.

**Void flag** **(int a ,int b, int bb,int d):**

This method used in add, sub, cmp methods. It changes all flags.

**Void mov** **(string a, string b, string c):**

This method controls all possibilities of mov command and make changes.

**void dec** **(string a):**

This method controls all possibilities of dec command and make changes.

**void inc** **(string a):**

This method controls all possibilities of inc command and make changes.

**void add(string a, string b,string c):**

This method controls all possibilities of add command and make changes.

**void sub(string a, string b,string c):**

This method controls all possibilities of sub command and make changes.

**void mul(string a):**

This method controls all possibilities of mul command and make changes.

**void div(string a):**

This method controls all possibilities of div command and make changes.

**int and2(int a, int b, int bit):**

This method helps and1 method. It change flags and return values.

**void and1(string a, string b,string c):**

This method controls all possibilities of and command and make changes.

**int or2(int a, int b, int bit):**

This method helps or1 method. It change flags and return values.

**void or1(string a, string b,string c):**

This method controls all possibilities of or command and make changes.

**int xor2(int a, int b, int bit):**

This method helps xor1 method. It change flags and return values.

**void xor1(string a, string b,string c):**

This method controls all possibilities of xor command and make changes.

**int not2(int a, int bit):**

This method helps not1 method. It change flags and return values.

**void not1(string a):**

This method controls all possibilities of not command and make changes.

**int shift2(int a,int count,int bit,int c):**

This method helps shift1 method. It change flags and return values.

**void shift1(string a, int count, int c):**

This method controls all possibilities of shl and shr commands and make changes.

**int rot2(int a, int count, int bit, int c):**

This method helps rot1 method. It change flags and return values.

**void rot1(string a, int count, int c):**

This method controls all possibilities of rcr and rcl commands and make changes.

**void push(string a):**

This method controls all possibilities of push command and make changes.

**void pop(string a):**

This method controls all possibilities of pop command and make changes.

**void cmp(string a, string b, string c):**

This method controls all possibilities of cmp command and make changes.