1) start

Start of the program. Makes cx and bx registers 0.

2) input

Reads input char by char. cx holds an integer's digits to obtain the whole integer. After that looks for the input char whether is space, newline or not. Then goes to dummy jump.

3) fin

Check is to prevent pushing into stack whenever the space is after an operation character. Then if it is not operator, then pushes the integer to stack.

4) lfin

If only one number is given in the input without any other operation or space. Then jumps to lfinonevalue. Other then that, it is the end of taking input since this is for newline. Pops the result to ax register. Uses number variable and bx register to put $ end of the string. Then goes to convert_hexadecimal.

5) lfinonevalue

Cx holds the result for one number input. Move cx to ax register. Uses number variable and bx register to put $ end of the string. Then goes to convert_hexadecimal.

6) bdec

Convert the digit to its real value by subtracting the ascii value. Then puts one digit to cx register. If there is more digits for the integer then "bdec" puts new digits to cx. However, if it is a new integer then cx comes as 0.

7) letter

Decreases the ascii code even more since letters' ascii value are not consecutive with digits'

8) lowercase

Decreases the ascii code even more since lowercase letters' ascii value are not consecutive with digits'.

9) dummyjump

For the controlling operations part makes check 0 to hold the information of the input if it is an operator. For the given input jumps to the corresponding name.

For the controlling number part makes check 1 to hold the information of the input if it is a number. For the given input jumps to the corresponding name.

10) badd

Pops 2 number from stack adds them and pushes it back

11) bdiv

Pops 2 number from stack divides the first one to the second one and pushes the quotient back

12) bmult

Pops 2 number from stack multpilies them and pushes the result back

13) bxor

Pops 2 number from stack does the bitwise xor operation and pushes the result back

14) bor

Pops 2 number from stack does the bitwise or operation and pushes the result back

15) band

Pops 2 number from stack does the bitwise and operation and pushes the result back

16) convert_hexadecimal

Converts the ax integer to 10h to have the right most digit of the number since ax holds the number. If it is bigger or equal to 10 then it will be converted to letters, so it goes to addlet. Otherwise it goes to adddec.

17) adddec:

Converts the digit to character again and stores it in the memory by using bx register. Bx holds the place in number array. It puts the digit into number array then decrement the bx and goes to convert_hexadecimal again. Bx now holds the place on one left. If the converting is finished, goes to printnl.

18) addlet:

Converts the letter to character again and stores it in the memory by using bx register. Bx holds the place in number array. It puts the letter into number array then decrement the bx and goes to convert_hexadecimal again. Bx now holds the place on one left. If the converting is finished, goes to printnl.

19) printnl:

Prints a new line and backspace character in order to make output more readable.

20) print:

If the pointer in the bx register is different from address of number jumps to printloop, otherwise prints the string that starts from the address that is kept in bx register.

21) printloop:

Prints a ‘0’, increments the address in cx register and then jumps back to print.

22) exit:

Concludes the program.