Computing Assignment #1

Simple Calculator

Documentation

Kamil Przepiórowski

Student no. 17327895

Part One- Console Input

This part of the program takes in input from the user, stores it in a register and prints the input back out in the console. The program follows this pseudo-code.

```
read input key from console
while (key != enter)
{
print key back in console
result=result x 10
convert value from ASCII
result=result+value entered
}
```

Unless the key entered is 'Enter' then the program loops around, multiplying the previous input value by 10, and adding on the next value entered to that product.

If a value is entered, which has more than 1 digit; e.g. 716. Then the program takes each digit individually and runs it through the program until each digit is processed and the input key is 'Enter'. The following would occur;

➔Program takes in the '**7**' 0x10=0

0+**7**=7

➔Program takes in the '**1**' 7x10=70

70+**1**=71

➔Program takes in the '**6**' 71x10=710

710+**6**=716

The final value printed back to the console is '716', and is stored in R4 as '0x2CC'.

Once the input value is the key 'Enter', the program stops taking any further inputs.

Input from user/Output printed in console	Value stored in register R4
512	0x200
b	0x32
A	0x11

Part Two- Expression Evaluation

In this part, the program takes in two numerical values inputted by the user, and one sign operand in between them, and performs the calculation, storing the result in a register, R5. The program follows the following pseudo-code.

```
read 1st number from console
while (key != '+')||(key!='-')||(key!='*') {
print key back in console
result= result x 10
convert value from ASCII
result=result+value }

print sign back in console

read 2nd number from console
while (key != enter){
print key back to console
result= result x 10
convert value from ASCII
result=result+value }

if(sign=='+') do firstNumber+secondNumber
else if(sign=='-') do firstNumber-secondNumber
else if(sign=='*') do firstNumber*secondNumber
else endProgram
```

The first number is processed and printed out in the same fashion as in Part One. However, the program stops taking input for that number once a sign operator '+', '-' or '*' is entered, which is then printed to the screen.

The second number follows the same path as Part One.

After the input from the user is complete, the program checks if the sign operand entered is a valid one, if so, it does the corresponding calculation, storing the answer in a separate register. If an invalid entry is input, the program is shut down and no value is stored as the result of the calculation.

Input from user/Output on console	Value stored in register R5
100+250	0x15E
14-6	0x08
12*5	0x3C
133%3	Value remains unchanged (0xA00003A0)

Part Three- Displaying the Result

This part of the program is responsible for printing out the answer of the calculation, obtained in the previous part, in decimal form.

To do this, the program divides the answer by the highest power of 10 that fits into the answer, prints the quotient, and then follows the same steps for the remainder of the division.

e.g. if the answer is '512'

The highest power of 10 that goes into 512 is 100.

512/100= 5 remainder 12

The 5 is printed out.

The highest power of 10 that goes into 12 is 10.

12/10= 1 remainder 2

The 1 is printed.

The highest power of 10 that goes into 2 is 1.

2/1= 2

The 2 is printed.

In the end, the answer '512' is printed back in the console.

To follow this process, the program first needs to calculate the number of digits in the answer of the calculation from part two. It does this by following this pseudo-code.

```
b=1;
quotient=0;
while((b!=0)&&(b<answerOfCalculation)){
quotient += 1;
b= b x 10;}

if(quotient=0)
quotient=1;
```

This makes sure the number of digits in the answer, is always stored as at least 1. By finding the number of digits in the answer of the calculations, we can find the highest power of 10 which divides into that answer.

This value is *'number of digits in the answer – 1'*

e.g.

If the answer is 512; it has 3 digits. Which means the highest power of 10 which divides into that value is 2, and 10^2 is 100.

➔(1!=0)&&(1<512)

Quotient=1 (0+1)

10=1 x 10

➔(10!=0)&&(10<512)

Quotient=2 (1+1)

100=10 x 10

➔(100!=0)&&(100<512)

Quotient=3 (2+1)

1000=100 x 10

➔(1000!=0)&&(1000!<512)

As 1000!<512, the counter stops, and stores that the value '512' has 3 digits.

This means that the program will divide '512' by 100 (10^2) in the next section, as this is the highest power of 10 that goes into the value of the answer.

After finding the number of digits in the answer of the calculation, the program then prints '=' and follows the example shown previously, looping around this pseudo-code until the full number is printed back on the console of the user.

```
highestPowerOf10=1;
amount of digits= amount of digits -1;
while(amountOfDigits!=0){
highestPowerOf10= higherstPowerOf10 x 10;
amount of digits= amount of digits -1;}
R0=0;
while((answerOfCalculation!=0)&&(answerOfCalculation>highestPowerOf10)){
digitToBePrinted= digitToBePrinted + 1;
answerofCalculation = answerOfCalculation-highestPowerOf10;}
convert digitToBePrinted to ASCII
print digitToBePrinted
```

Before each ‘BL sendchar’ instruction, the value stored in R0 is converted back to ASCII so that it can be printed on the screen of the user in decimal form.

After each ‘BL sendchar’ instruction, the program takes 1 away from the number of remaining digits, which in turn lowers the highest power of 10.

Input from user	Value stored in register R11	Output on console
365+34	0x18F	365+34=399
512+512	0x400	512+512=1024
0+0	0x00	0+0=0
124+0	0x7C	124+0=124
15-9	0x06	15-9=6
36-36	0x00	36-36=0
0-0	0x00	0-0=0
1567-0	0x61F	1567-0=1567
27*9	0xF3	27*9=243
578*0	0x00	578*0=0
0*0	0x00	0*0=0