שנקר - בי“ס גבוה להנדסה ולעיצוב
רחוב אנה פרנק 12, רמת גן 52526
טל.: 03-6110000 פקס: 03-7521141
דוא“ל: info@shenkar.ac.il

Software Engineering Department

Computer Organisation and Programming Course
final assignment

Pocket Calculator application

Written by:

Noor Haj Dawood   314997602

04 July 2020

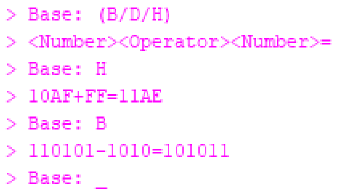
Lecturer: Dr Yigal Hoffner

TABLE OF CONTENTS

1 Pocket Calculator application design

The calculator interface looks like this:

- Base: (B/D/H)
- <Number><Op><Number>=*Result*
- A flashing '_' to indicate that input is waiting



```
> Base: (B/D/H)
> <Number><Operator><Number>=
> Base: H
> 10AF+FF=11AE
> Base: B
> 110101-1010=101011
> Base: _
```

1.1 Extra Work Carried Out (extra 25% items)

- 5% for special input/output
  1) Input Base: (B/D/H), or X to terminate
  2) Input number1 followed by operator followed by number2
  3) Input '=' right after to execute
  - If an illegal base is keyed in, an error will pop out
  - If there's an overflow while input the numbers an error will pop out
  - If while inputting numbers an illegal digit is keyed in an error will pop out
  - If an illegal operator keyed in and error will pop out
- 5% for Hexadecimal and Binary input/output
  Input numbers are in the chosen base as will as the output
- 10% for fast division and fast multiplication
  - I used table multiplication to implement fast multiplication
  - I used long division for binary numbers to implement fast division
- 10% for sophisticated input
  - The input is formatted as mentioned above: (no need to press 'CR' or space)
    - <base>
    - <number><op><number>=

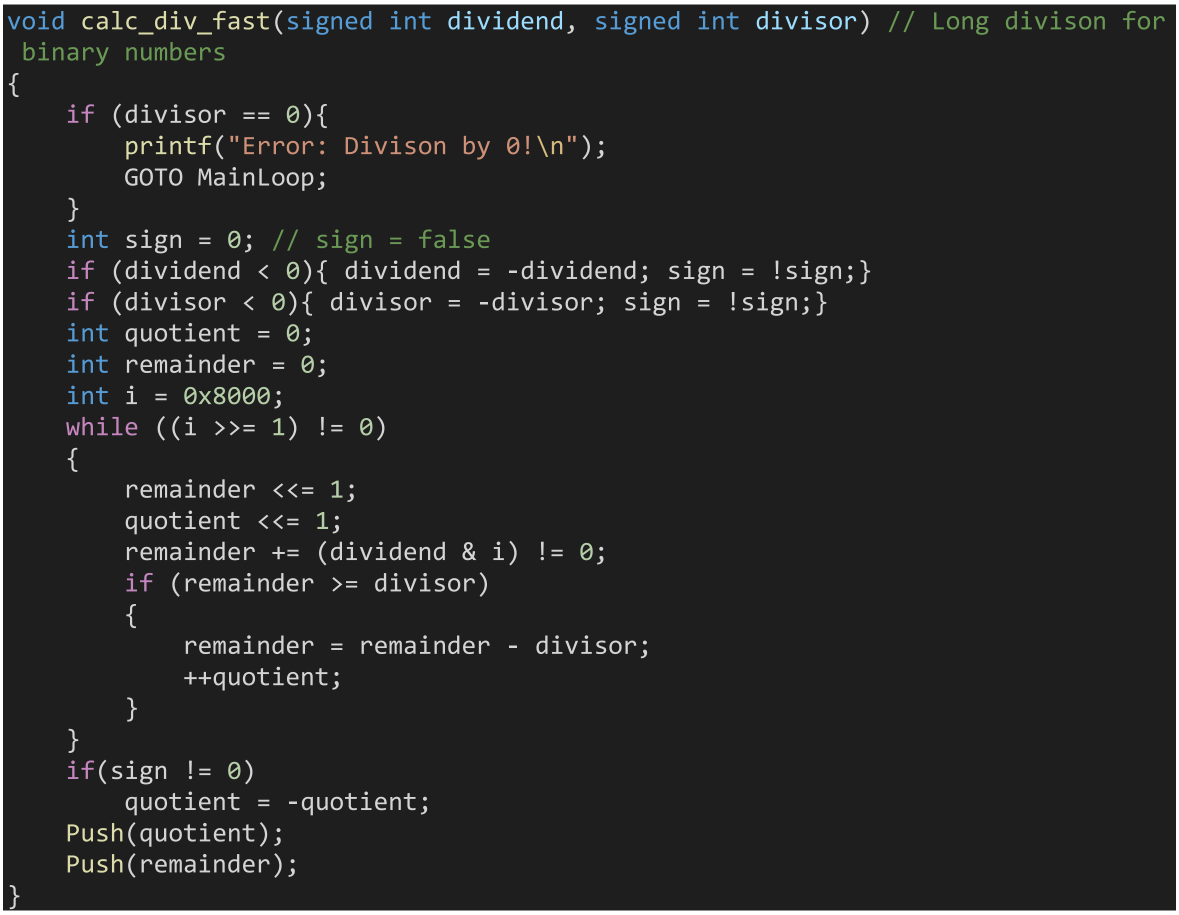
1.2 Major Design/Implementation Decisions

The following major design decisions were made during the design and implementation phases:

- Integers are represented internally using the 2's complement representation
- Parameters to the subroutines are passed through the Accumulator if it's one parameter at a time, and for multiple parameters they are passed through the stack (or mixed)
- Parameters returning from the subroutines are passed through Accumulator if it's one parameter at a time, or through the stack for multiple parameters (or mixed)
- After a Error occurs, an appropriate subroutine for printing the error is called, and then return to the start of the main calculator loop

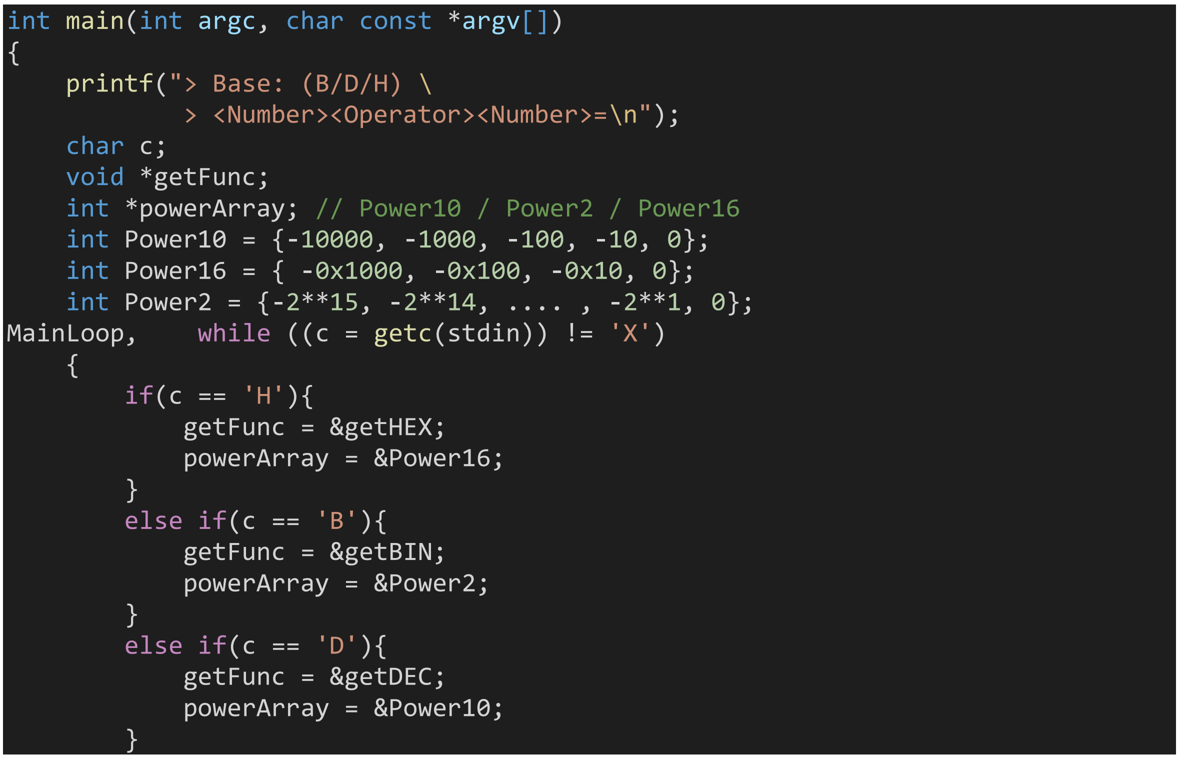
1.3 The High-level Algorithms

Fast Division Implementation: (Long Division)



```
void calc_div_fast(signed int dividend, signed int divisor) // Long divison for
 binary numbers
{
    if (divisor == 0){
        printf("Error: Divison by 0!\n");
        GOTO MainLoop;
    }
    int sign = 0; // sign = false
    if (dividend < 0){ dividend = -dividend; sign = !sign;}
    if (divisor < 0){ divisor = -divisor; sign = !sign;}
    int quotient = 0;
    int remainder = 0;
    int i = 0x8000;
    while ((i >>= 1) != 0)
    {
        remainder <<= 1;
        quotient <<= 1;
        remainder += (dividend & i) != 0;
        if (remainder >= divisor)
        {
            remainder = remainder - divisor;
            ++quotient;
        }
    }
    if(sign != 0)
        quotient = -quotient;
    Push(quotient);
    Push(remainder);
}
```

Main Implementation:



```
int main(int argc, char const *argv[])
{
    printf("> Base: (B/D/H) \
            > <Number><Operator><Number>=\n");
    char c;
    void *getFunc;
    int *powerArray; // Power10 / Power2 / Power16
    int Power10 = {-10000, -1000, -100, -10, 0};
    int Power16 = { -0x1000, -0x100, -0x10, 0};
    int Power2 = {-2**15, -2**14, .... , -2**1, 0};
MainLoop,    while ((c = getc(stdin)) != 'X')
    {
        if(c == 'H'){
            getFunc = &getHEX;
            powerArray = &Power16;
        }
        else if(c == 'B'){
            getFunc = &getBIN;
            powerArray = &Power2;
        }
        else if(c == 'D'){
            getFunc = &getDEC;
            powerArray = &Power10;
        }
        else{
            printf("Error: no such base!\n");
            goto MainLoop
        }

        int num1 = getNumber(getFunc);
        char op = getc(stdin);
        if (checkOP(op) == 0){
            printf("Error: %c is not an operator!\n", op);
            goto MainLoop;
        }
        int num2 = getNumber(getFunc);
        int result, remainder;
        if(op == '+'){
            result = num1 + num2;
            // check over/underflow using MSB of num1 and num2 and result, like
 we did in the homework
            printf("%d\n", result);
        }
        else if(op == '-'){
            result = num1 - num2;
            // check over/underflow using MSB of num1 and num2 and result, like
 we did in the homework
            printf("%d\n", result);
        }
        else if(op == '*'){
            result = calc_mult_fast(num1, num2);
            printf("%d\n", result);
        }
        else // op == '/'
        {
            calc_div_fast(num1, num2);
            remainder = Pop_AC();
            result = Pop_AC();
            printf("%d",result);
            if(remainder != 0){
                printf("(%d)", remainder);
            }
            printf("\n");
        }
    }
    printf("Bye ^_^");
    return 0;
}
```

The rest of the subroutines:

Push_AC

Push AC to the stack

Pop_AC

Pop top of stack to AC

OpCheck

Check if AC is + - * /, if yes return AC else return 0 (FALSE)

check09

Check if AC is 09, if yes return AC else return 0 (FALSE)

calc_mult

Table multiplication algorithm using shifts bitwise

calc_div

Long Division using shifts bitwise (C code above)

Putc

Print AC to display

Getc

Input character from v-keyboard (flashes _ to indicate it’s waiting for input)

Puts

Prints Array to display

Gets

Inputs to Array from v-keyboard, stops at ‘CR’

putSignedInteger

Outputs signed number to display using the correct base chosen by the user

GetSignedInteger

Inputs signed number from v-keyboard using the correct base chosen by the user

digError

prints that an illegal digit was inserted and restarts the calculator

ouf

prints that over/underflow has happened and restarts the calculator

checkAF

check if AC is ‘A’ to ‘F’ , if yes return AC else return 0 (FALSE)

check01

check if AC is ‘0’ or ‘1’, if yes return AC else return 0 (FALSE)

getHEX

input unsigned HEX number from v-keyboard

getBIN

input unsigned BIN number from v-keyboard

getDEC

input unsigned DEC number from v-keyboard

2 The User Guide

The calculator interface looks like this:

- ➢ Base: (B/D/H)
- ➢ <Number><Op><Number>=*Result*

- A flashing '_' indicates input is ready
- In place of (B/D/H) input B for Binary, D for Decimal and H for Hexadecimal or X to terminate
- After that input the first number followed by an operator followed by the second number and then input the character '=' to execute.

- Input X as a base to terminate the calculator
- Number digits for Binary are 0 or 1
- Number digits for Decimal are 0 to 9
- Number digits for Hexadecimal are 0 to 9 or A to F (A to F are capital letters)
- Input '=' after inputing the second number to execute

Error messages will appear for the following cases:

1) An illegal base
2) An illegal digit for the chosen base
3) Input number bigger than 32767(Decmial) or smaller than -32768(Decimal)
4) An illegal operator
5) Over/underflow after executing the operator

After an error message the calculator will start a new input sequence