Lab 4 - Mathematics

Dr. Donald Davendra
CS311 - Computer Architecture 1

October 25, 2024

The fourth laboratory exercise requires you to assign the contents of one array and one variable in **nasm** and calculate the result based on the following equation (1):

$$\frac{\prod_{i=0}^{N-1}(x_i)}{y} \tag{1}$$

where N is the size of the array x.

Create a file named `math.asm` in ebe.

Question 1 - `.data` section.

You are required to assign one array and one variable in the `.data` segment as the following:

- label - `x`, `y`
- contents - `x` = {-2, 4, 5} and `y` = -6
- size - `x` (**byte**) and `y` (**byte**)

The `segment .data` is given as:

```
     segment .data
x          db   -2, 4, 5     ; array of 3 values
y          db   -6           ; variable
quot       dq    0           ; quotient
rem        dq    0           ; remainder
```

You can declare other variables as you deem necessary to solve this assignment.

Question 2 - `.text` section.

Start the text segment as the following:

```
          segment .text
          global  main
     main:
```

Question 3 - `global main` section.

The task in the main section is to **explicitly** follow the equation, without any numeric simplification.

- You can only use the **1 operand** `imul` and `idiv` opcodes.
- After the final division operation, move the quotient to `quot` memory location and the remainder to `rem` memory location.
- If the quotient is positive, move it to the r8 register.
- If the quotient is negative, move it to the r9 register.

You are allowed to use a maximum of **three general purpose registers** in this lab in addition to r8 and r9 registers (which should only be used in the end of the code to store final values). You are NOT allowed to change any values in ***x*** and ***y***. Some of the opcodes of use in this lab are:

- `mov` - moving data from register-register, register-variable etc
- `lea` - loading effective address of a variable to a register.
- `imul` - multiply two signed values in registers.
- `idiv` - divide signed value
- `cqo` - instruction (available in 64-bit mode only) copies the sign (bit 63) of the value in the RAX register into every bit position in the RDX register
- `cmov` - conditional move opcodes
- `test` - clears the flags CF and OF to zero. The SF is set to the most significant bit of the result of the AND operartion. If the result is 0, the ZF is set to 1, otherwise set to 0

Upon completion of the task, zero out all the used registers and return. This following can be taken as an example:

```
          ...                     ; your code
          xor     rax, rax        ; zero out rax
          ret
```

Submission

Only submit the `math.asm` file to Canvas. All submitted files **MUST** have the **student name**, **student CWU ID** and the **honor code**.

The file must be submitted through Canvas before 5pm, Nov 1, 2024. The grading rubric is given in Table 1.

Table 1: Grading rubric

File	Aspects	Points
`math.asm`	Correct equation interpretation	10
	Correct use of registers	20
	Compiles and correct result	15
	Correct multiplication and division operands	20
	Status flag check and move to r8 - r9 register	20
	Documentation and Requirements	15