SEE2030: Introduction to Computer Systems (Fall 2016)

Programming Assignment #2:

64-bit arithmetic using 32-bit integers in assembly language

Due: Nov. 13th (Sunday), 11:59PM

1. Introduction

The purpose of this assignment is to become familiar with the x86-64 instructions and to understand how to program in assembly language.

2. Problem specification

2.1. Overview

Write assembly codes for the `Uadd64()`, `Usub64()`, and `Umul64()` functions you've implemented in PA#1. As in PA#1, each function receives two 64-bit integers and computes the addition, subtraction, and multiplication of those integers, respectively. The prototypes of each functions are as follows:

```
typedef struct {
     u32 hi;
     u32 lo;
} HL64;
```

```
HL64 Uadd64 (HL64 a, HL64 b);
HL64 Usub64 (HL64 a, HL64 b);
HL64 Umul64 (HL64 a, HL64 b);
```

The `HL64` type is the alias of a structure which holds high 32 bits and low 32 bits of a single 64-bit integer. Two arguments, `a` and `b`, represent the operands. The return value should store upper 32 bits and lower 32 bits of the 64-bit result. The `u32` type is the alias of `unsigned int` type.

2.2. Backgrounds

The function arguments `a` and `b` are described in HL64 type. In x86-64 machine, these arguments are stored in `%rdi` and `%rsi` registers, respectively. Figure 1 shows the organization of function arguments a, b, and return value x. The high 32 bits are for lo, and the low 32 bits are for hi. The return value of each function is also HL64 type. Therefore, the high 32 bits of `%rax` (lo) should contain low 32 bits of result integer and low 32 bits of `%rax` (hi) should contain high 32 bits of result integer.

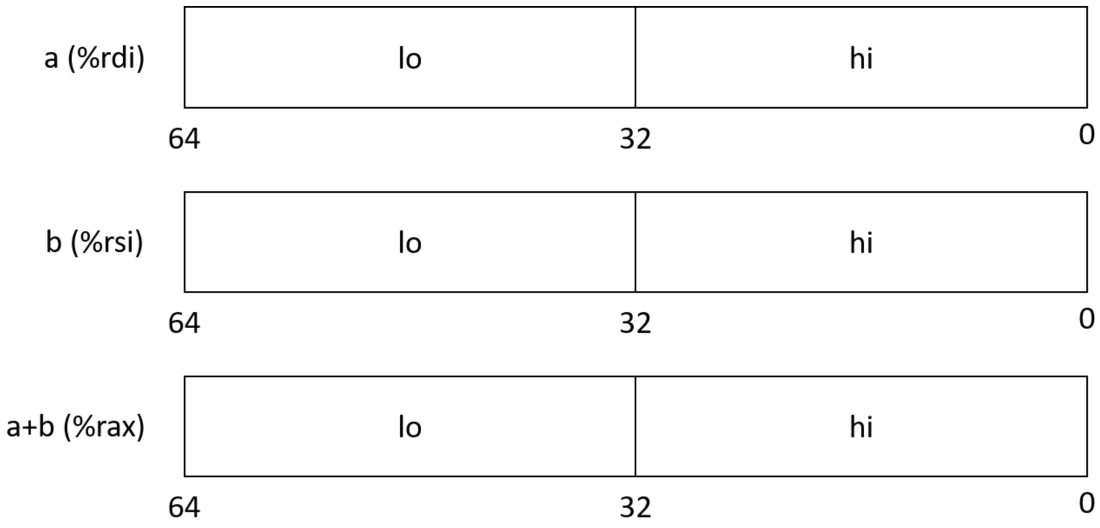


Figure 1. Organization of function arguments and return value

2.3. Restrictions

(1) For computation, you should use only 32-bit arithmetic & logical instructions, such as `cmpl`, `testl`, `addl`, `subl`, `mull`, `sall`, `sarl`, `shrl`, `shll`, `xorl`, `andl`, `orl`, `incl`, `decl`, `negl`, `notl`, etc. You can freely use `movl`, `movq`, or `cmov` (conditional move) instructions to transfer data. You are also allowed to use any conditional branch instructions (`je`, `jne`, `ja`, `jae`, `jb`, `jbe`, etc.).

(2) You should use only the following registers: `%rdi`, `%rsi`, `%rax`, `%rdx`, `%rcx`, and their 32-bit registers such as `%edi`, `%esi`, `%eax`, `%edx`, `%ecx`.

If you violate the above restrictions, you will get no points.

2.4. Verification of your result

Since a "`long long`"-type integer consists of 64 bits in 32-bit machines, another way to obtain the true result of operations of 64-bit integers just operates two `long long` type variables, as shown in the following code example.

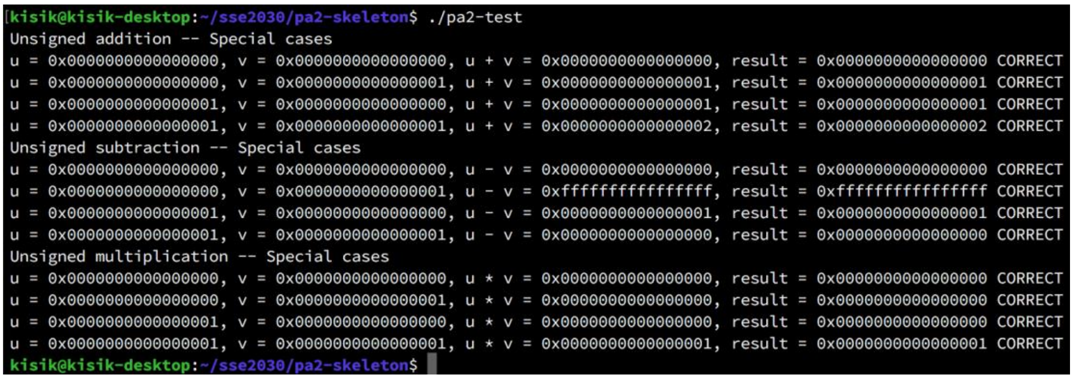
```
long long x, y, z;
z = x + y;
```

Therefore, your result should be identical to the result obtained by the above code example. More specifically, if the return structure of functions is `x`, `x.hi` should have the value of `(int) (z >> 32)` and `x.lo` should have the value of `(int) (z & 0xffffffff)`.

3. Example

The skeleton of program attached (pa2-test.c).

Some sample runs:



```
[kisik@kisik-desktop:~/sse2030/pa2-skeleton$ ./pa2-test
Unsigned addition -- Special cases
u = 0x0000000000000000, v = 0x0000000000000000, u + v = 0x0000000000000000, result = 0x0000000000000000 CORRECT
u = 0x0000000000000000, v = 0x0000000000000001, u + v = 0x0000000000000001, result = 0x0000000000000001 CORRECT
u = 0x0000000000000001, v = 0x0000000000000000, u + v = 0x0000000000000001, result = 0x0000000000000001 CORRECT
u = 0x0000000000000001, v = 0x0000000000000001, u + v = 0x0000000000000002, result = 0x0000000000000002 CORRECT
Unsigned subtraction -- Special cases
u = 0x0000000000000000, v = 0x0000000000000000, u - v = 0x0000000000000000, result = 0x0000000000000000 CORRECT
u = 0x0000000000000000, v = 0x0000000000000001, u - v = 0xffffffffffffffff, result = 0xffffffffffffffff CORRECT
u = 0x0000000000000001, v = 0x0000000000000000, u - v = 0x0000000000000001, result = 0x0000000000000001 CORRECT
u = 0x0000000000000001, v = 0x0000000000000001, u - v = 0x0000000000000000, result = 0x0000000000000000 CORRECT
Unsigned multiplication -- Special cases
u = 0x0000000000000000, v = 0x0000000000000000, u * v = 0x0000000000000000, result = 0x0000000000000000 CORRECT
u = 0x0000000000000000, v = 0x0000000000000001, u * v = 0x0000000000000000, result = 0x0000000000000000 CORRECT
u = 0x0000000000000001, v = 0x0000000000000000, u * v = 0x0000000000000000, result = 0x0000000000000000 CORRECT
u = 0x0000000000000001, v = 0x0000000000000001, u * v = 0x0000000000000001, result = 0x0000000000000001 CORRECT
kisik@kisik-desktop:~/sse2030/pa2-skeleton$
```

4. Hand in instructions

- Submit only pa2.s file to the submission site (http://sys.skku.edu).

5. Logistics

- You will work on this assignment alone.
- Only the assignments submitted before the deadline will receive the full credit. 25% of the credit will be deducted for every single day delay.
- Any attempt to copy others' work will result in heavy penalty (for both the copier and the originator). Don't take a risk.

Good luck!