SEE2030: Introduction to Computer Systems (Fall 2016)

Programming Assignment #1:

64-bit arithmetic using 32-bit integers

Due: October 4th, 11:59PM

1. Introduction

The purpose of this assignment is to become more familiar with the binary representation of integers and to understand what happens during the operation of two integers.

2. Problem specification

2.1. Overview

Write three C functions named `Uadd64()`, `Usub64()`, and `Umul64()` which receive two 64-bit integers and compute the addition, subtraction, and multiplication of those integers, respectively. Note that we use two 32-bit integers to represent a 64-bit integer. The prototype of each function is 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 32bits and low 32bits of a single 64-bit integer. Two arguments, `a` and `b`, represent the operands. The return value should store upper 32bits and lower 32bits of result 64-bit integer.

The `u32` type is the alias of `unsigned int` type.

2.2. Restrictions

You should use only `int` and `HL64` type variables. In addition, you are allowed to use only integer arithmetic and logical operations inside `Uadd64(), Usub64, and Umul64()` functions.

2.3. Verification of your result

Since the "unsigned `long long (u64)`" type represents unsigned 64-bit integers, another way to verify your result is to perform the same computation using this type of variables. In the `pa1.h` file, we provide two macros called `U64_TO_HL64(u,x)` and `HL64_TO_U64(x,u)`, which convert a "`u64`"-type variable to and from the corresponding "`HL6`"-type variable, respectively. Using these macro, you can check whether your computation result is correct or not as shown in the following example. For the given u64-type variables `u` and `v`, the result of `Real_Uadd64(u, v)` should be identical to that of `HL_Uadd64(u, v)`.

```
#define U64_TO_HL64(u,x) (x).hi = (u32) ((u64) (u) >> 32), \
                         (x).lo = (u32) ((u64) (u) & 0xffffffff)
#define HL64_TO_U64(x,u) (u) = (((u64) (x).hi << 32) | (u64) x.lo)

u64 Real_Uadd64 (u64 u, u64 v)
{
     return u + v;
}

u64 HL_Uadd64 (u64 u, u64 v)
{
     HL64 a, b, x;
     U64_TO_HL64 (u, a);
     U64_TO_HL64 (v, b);
     x = Uadd64 (a, b);          // Your implementation
     HL64_TO_U64 (x, result);
     return result;
}
```

3. Example

The test code of this assignment is available in the **"`pa1-test.c`"** file.

Some sample runs:

```
kisik@kisik-desktop:~/sse2030/pa1$ ./pa1-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/pa1$
```

4. Hand in instructions

- Submit only **"`pa1.c`"** 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!