



4190.308: Computer Architecture (Fall 2018)

Project #1: 64-bit Arithmetic using 32-bit Integers

Due: September 30th (Sunday), 11:59PM

1. Introduction

The purpose of this project is to become more familiar with the binary representation of integers and to understand what happens during the arithmetic operations between 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 unsigned int u32;  
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 the result separately. The `u32` type is the alias of `unsigned int` type.

2.2 Restrictions

- You should use only (signed or unsigned) int- and HL64-type variables.
- You are allowed to use only integer arithmetic and logical operations inside `Uadd64()`, `Usub64`, and `Umul64()` functions.
- Do not use any array 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 “HL64”-type variable, respectively. Using these macros, 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 result;  
  
    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 project is available in the "pa1-test.c" file.

Some sample runs:



```

@ sys
$ make
gcc -g -O2 -Wall -c pa1-test.c -o pa1-test.o
gcc -g -O2 -Wall -c pa1.c -o pa1.o
gcc -g -O2 -Wall -o pa1-test pa1-test.o pa1.o
$ ./pa1-test
Unsigned addition:
u = 0x45777b23e63c9869, v = 0x4877b0dc59495cff, u + v = 0x8def2c003f85f568, result = 0x0000000000000000 WRONG
u = 0x946a5558ef8e1f29, v = 0x7cfd1b58fa7ed7ab, u + v = 0x116770b1ea0cf6d4, result = 0x0000000000000000 WRONG
u = 0x41f3b71efbe2a9e3, v = 0xe1575f007fd062c2, u + v = 0x234b161f7bb30ca5, result = 0x0000000000000000 WRONG
u = 0x085db127fa16231b, v = 0xe9f990cde7ef438d, u + v = 0xf25741f5e20566a8, result = 0x0000000000000000 WRONG
Unsigned subtraction:
u = 0x0f7752255a9cf92e, v = 0x727fdcc23befd79f, u - v = 0x9cf775631ead218f, result = 0x0000000000000000 WRONG
u = 0xc4eb6807de6afb66, v = 0x5d739b504f1bd7b7, u - v = 0x6777ccb78f4f23af, result = 0x0000000000000000 WRONG
u = 0xba7d83e47d7130a3, v = 0xd95b6c61678c895d, u - v = 0xe122178315e4a746, result = 0x0000000000000000 WRONG
u = 0xb1771da33743a858, v = 0x5aef63847fa2a8d4, u - v = 0x5687ba1eb7a0ff84, result = 0x0000000000000000 WRONG
Unsigned multiplication:
u = 0xbd7b838cf353d0cd, v = 0xe0de9a76dfe49eb4, u * v = 0xc78bc74f5b615624, result = 0x0000000000000000 WRONG
u = 0x247ca8861936c40e, v = 0x1dba95f87c138641, u * v = 0x4f5d74d666911b8e, result = 0x0000000000000000 WRONG
u = 0xf57d3dbd7f7b8ddc, v = 0xf0a7221a7516dde9, u * v = 0x9cbdd5526dd3093c, result = 0x0000000000000000 WRONG
u = 0xc87f4fd4e19ac241, v = 0xf8e50badfd072367, u * v = 0x7cfe67a17a9a0b27, result = 0x0000000000000000 WRONG
$

```

4. Required setups for this and future project assignments

4.1 Installing Linux

Your C code should work after compiling it with gcc (GNU C Compiler) on Linux.

The official Linux platform in this course is Ubuntu 18.04.1 LTS which can be downloaded from <https://www.ubuntu.com>. You can install Ubuntu on the Windows machine by using virtualization products such as Oracle VirtualBox (<https://www.virtualbox.org>) or VMware Workstation Player (<https://www.vmware.com>). For further details, please google them.

4.2 Creating an account in the submission server

Register your account in the submission server <https://sys.snu.ac.kr>. **You must enter your real name & student ID (20XX-YYYYY format)**. You can see the project page after we approve your account.

4.3 Sending an email for remote access

Currently, the submission server is configured to be accessible only from SNU Campus network (147.46.*.* and 147.47.*.* IP addresses). If you want to submit your code outside of the campus, please let me know your IP address by sending an email to jinsoo.kim@snu.ac.kr. **This should be done at least 24 hours before your submission!** (Use <http://ip-address.us> to get your IP address)



SNU Systems Software & Architecture Lab.

Login

Email

Password

We recommend the Google Chrome web browser.
© SNU Systems Software & Architecture Laboratory

SNU Systems Software & Architecture Lab.

Registration

Email

Password

Password Confirm

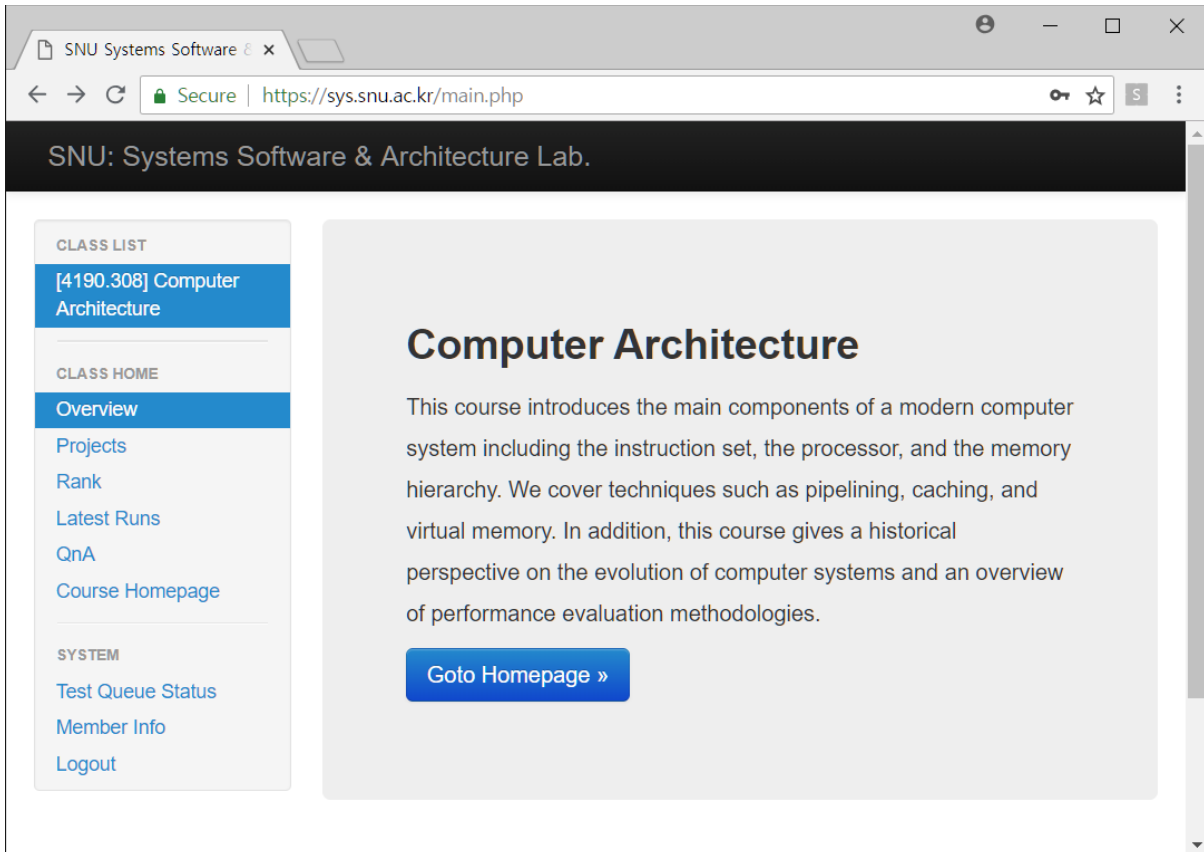
Name

NickName

Student No.

Mobile - -

We recommend the Google Chrome web browser.
© SNU Systems Software & Architecture Laboratory



The screenshot shows a web browser window with the URL <https://sys.snu.ac.kr/main.php>. The page title is "SNU: Systems Software & Architecture Lab.". On the left, there is a sidebar menu with sections: "CLASS LIST" (containing "[4190.308] Computer Architecture"), "CLASS HOME" (containing "Overview", "Projects", "Rank", "Latest Runs", "QnA", "Course Homepage"), and "SYSTEM" (containing "Test Queue Status", "Member Info", "Logout"). The main content area features the heading "Computer Architecture" and a paragraph: "This course introduces the main components of a modern computer system including the instruction set, the processor, and the memory hierarchy. We cover techniques such as pipelining, caching, and virtual memory. In addition, this course gives a historical perspective on the evolution of computer systems and an overview of performance evaluation methodologies." Below the text is a blue button labeled "Goto Homepage »".

5. Hand in instructions

- Submit only the **pa1.c** file to the submission server.

6. Logistics

- You will work on this project alone.
- Only the upload submitted before the deadline will receive the full credit. 25% of the credit will be deducted for every single day delay.
- You can use up to 5 *slip days* during this semester. Please let us know the number of slip days you want to use after each submission.
- Any attempt to copy others' work will result in heavy penalty (for both the copier and the originator). Don't take a risk.



서울대학교
SEOUL NATIONAL UNIVERSITY

Systems Software & Architecture Laboratory
Dept. of Computer Science and Engineering

Good luck!

Jin-Soo Kim

Systems Software & Architecture Laboratory

Dept. of Computer Science and Engineering

Seoul National University