

CPEN 411: Assignment 1  
Instrumentation, Program Analysis, and Modelling

Prashant J. Nair

September 11, 2024

# Contents

<b>1</b>	<b>Introduction and Setup [0 Points]</b>	<b>2</b>
1.1	Preparatory Steps: Setup . . . . .	2
1.2	Preparatory Steps: Algorithm . . . . .	3
1.3	Preparatory Steps: Pintool . . . . .	5
<b>2</b>	<b>Evaluation: Algorithm &amp; Pintool [5 Points]</b>	<b>9</b>
<b>3</b>	<b>Submission Instructions</b>	<b>11</b>

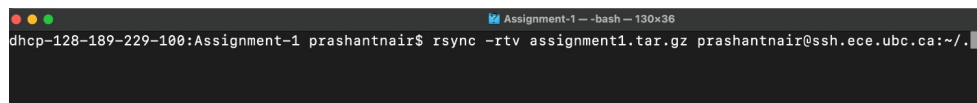
# Chapter 1

## Introduction and Setup [0 Points]

This assignment discusses program analysis using Pintool [1,2] and writing efficient searching algorithms. The assignment is divided into three parts. The first part looks to prepare you to start with this assignment. The second part looks at the assignment itself. The third part deals with the submission process. **Please read this document very carefully and in detail!**

### 1.1. Preparatory Steps: Setup

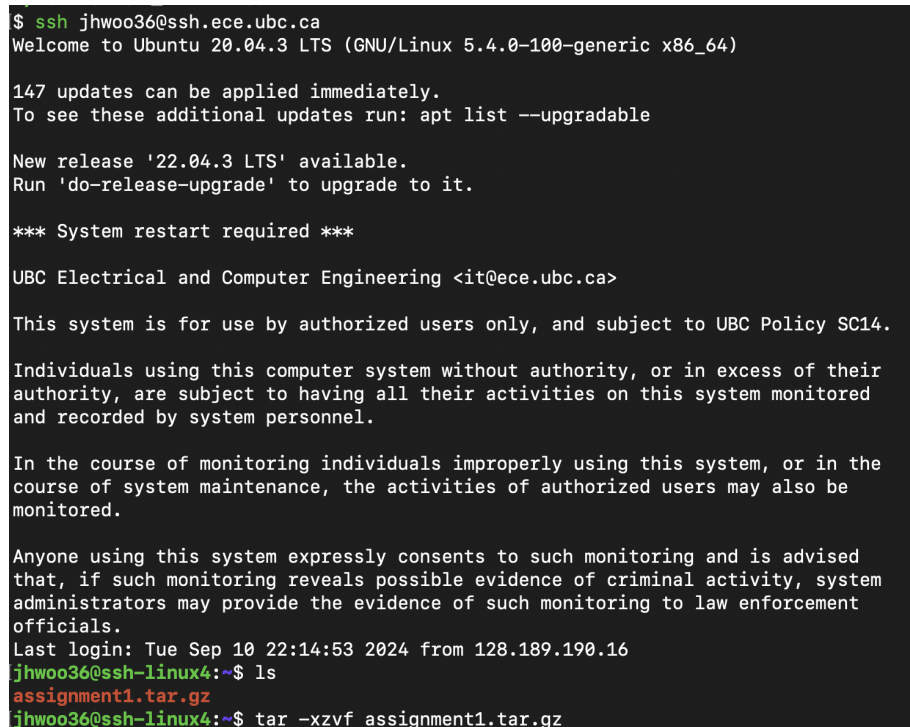
1. Copy the **assignment1.tar.gz** to the `<username>@ssh.ece.ubc.ca` server.



```
dhcp-128-189-229-100:Assignment-1 prashantnair$ rsync -rtv assignment1.tar.gz prashantnair@ssh.ece.ubc.ca:~/.
```

Suppose you do not have `rsync` (command) installed. You could try installing it. Alternatively, you can try using the `scp` command. For instance, Mac users can install these tools using Homebrew [3]. Windows and Mac users can use GUI tools like VSCode [4], CyberDuck [5], and MobaXterm [6] (**Windows Only**) to transfer files to remote servers. Linux users can use their *package manager* and install *rsync*, *scp*, or *sftp*.

2. `untar assignment1.tar.gz` in the ECE server.



```
$ ssh jhwoo36@ssh.ece.ubc.ca
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-100-generic x86_64)

147 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***

UBC Electrical and Computer Engineering <it@ece.ubc.ca>

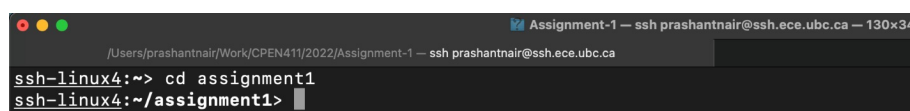
This system is for use by authorized users only, and subject to UBC Policy SC14.

Individuals using this computer system without authority, or in excess of their
authority, are subject to having all their activities on this system monitored
and recorded by system personnel.

In the course of monitoring individuals improperly using this system, or in the
course of system maintenance, the activities of authorized users may also be
monitored.

Anyone using this system expressly consents to such monitoring and is advised
that, if such monitoring reveals possible evidence of criminal activity, system
administrators may provide the evidence of such monitoring to law enforcement
officials.
Last login: Tue Sep 10 22:14:53 2024 from 128.189.190.16
jhwoo36@ssh-linux4:~$ ls
assignment1.tar.gz
jhwoo36@ssh-linux4:~$ tar -xzf assignment1.tar.gz
```

3. Change the directory into the **assignment1** folder.



```
ssh-linux4:~> cd assignment1
ssh-linux4:~/assignment1>
```

## 1.2. Preparatory Steps: Algorithm

1. Change directory into **assignment1/algorithms** folder.

```
ssh-linux4:~> cd assignment1
ssh-linux4:~/assignment1> cd algorithms
ssh-linux4:~/assignment1/algorithms>
```

2. Understand the searching algorithm in **searchoriginal.c** and **searchoriginal.h**

```
jhw0036@ssh-linux3: ~/assignment1/algorithms
jhw0036@ssh-linux3:~/assignment1/algorithms$ ls
assignment1.o Makefile README runall.sh searchnew.c searchnew.h searchoriginal.c searchoriginal.h
jhw0036@ssh-linux3:~/assignment1/algorithms$ vim searchoriginal.c
```

```
#include "searchoriginal.h"

long long int searchorig(unsigned long long int *arr, long long int size, long long int value){
    return linearSearch(arr,size,value); //Return the index of the element in arr[] which has data that is equal to value
}

long long int linearSearch(unsigned long long int *arr, long long int size, long long int value){
    unsigned long long int counter1, counter2;
    for (counter1 = 0; counter1 < size-1; counter1++){
        if (arr[counter1] == value){
            return counter1;
        }
    }
    return -1;
}
```

```
jhw0036@ssh-linux3: ~/assignment1/algorithms
jhw0036@ssh-linux3:~/assignment1/algorithms$ ls
assignment1.o Makefile README runall.sh searchnew.c searchnew.h searchoriginal.c searchoriginal.h
jhw0036@ssh-linux3:~/assignment1/algorithms$ vim searchoriginal.c
jhw0036@ssh-linux3:~/assignment1/algorithms$ vim searchoriginal.h
```

```
long long int searchorig(unsigned long long int *arr, long long int size, long long int value);
long long int linearSearch(unsigned long long int *arr, long long int size, long long int value);
```

3. Compile the source files in the **assignment1/algorithms** folder.

```
jhw0036@ssh-linux3: ~/assignment1/algorithms
jhw0036@ssh-linux3:~/assignment1/algorithms$ make
gcc -c searchnew.c
gcc -c searchoriginal.c
gcc -O3 assignment1.o searchnew.o searchoriginal.o -o assignment1.bin
jhw0036@ssh-linux3:~/assignment1/algorithms$ ls
assignment1.bin Makefile runall.sh searchnew.h searchoriginal.c searchoriginal.o
assignment1.o README searchnew.c searchnew.o searchoriginal.h
jhw0036@ssh-linux3:~/assignment1/algorithms$
```

4. Execute your compiled code.

```
ssh-linux4:~/assignment1/algorithms> bash runall.sh Your Student ID Number
```

5. Change to the **assignment1/output**
6. Make sure that **searchorig.txt** has PASSED.

```
jhw0036@ssh-linux3: ~/assignment1/output

jhw0036@ssh-linux3:~/assignment1/algorithms$ make
gcc -c searchnew.c
gcc -c searchoriginal.c
gcc -O3 assignment1.o searchnew.o searchoriginal.o -o assignment1.bin
jhw0036@ssh-linux3:~/assignment1/algorithms$ ls
assignment1.bin  Makefile  runall.sh  searchnew.h  searchoriginal.c  searchoriginal.o
assignment1.o    README   searchnew.c  searchnew.o  searchoriginal.h
jhw0036@ssh-linux3:~/assignment1/algorithms$ bash runall.sh
assignment1.bin  Makefile  runall.sh  searchnew.h  searchoriginal.c  searchoriginal.o
assignment1.o    README   searchnew.c  searchnew.o  searchoriginal.h
jhw0036@ssh-linux3:~/assignment1/algorithms$ bash runall.sh 100
jhw0036@ssh-linux3:~/assignment1/algorithms$ ls
assignment1.bin  Makefile  runall.sh  searchnew.h  searchnew.o  searchoriginal.h  searchoriginal.o
assignment1.o    README   searchnew.c  searchnew.inst  searchoriginal.c  searchoriginal.inst
jhw0036@ssh-linux3:~/assignment1/algorithms$ cd ../output/
jhw0036@ssh-linux3:~/assignment1/output$ ls
searchnew.txt  searchorig.txt

jhw0036@ssh-linux3:~/assignment1/output$ vim searchorig.txt

PASSED

prashantnair — ssh prashantnair@ssh.ece.ubc.ca — 130x34
ssh-linux4:~/assignment1/output> cd ../algorithms/
ssh-linux4:~/assignment1/algorithms>
```

7. Change to the `assignment1/algorithms`
8. Understand the x86-ISA instructions used in the **Linear-Search** algorithm

```
jhw0036@ssh-linux3: ~/assignment1/algorithms

jhw0036@ssh-linux3:~/assignment1/algorithms$ vim searchoriginal.inst
```

```
jhw0036

Disassembly of section .text:

0000000000000000 <searchorig>:
0:  f3 0f 1e fa          endbr64
4:  55                   push  rbp
5:  48 89 e5             mov   rbp,rsi
8:  48 83 ec 20          sub   rsp,0x20
c:  48 89 7d f8          mov   QWORD PTR [rbp-0x8],rdi
10: 48 89 75 f0          mov   QWORD PTR [rbp-0x10],rsi
14: 48 89 55 e8          mov   QWORD PTR [rbp-0x18],rdx
18: 48 8b 55 e8          mov   rdx,QWORD PTR [rbp-0x18]
1c: 48 8b 4d f0          mov   rcx,QWORD PTR [rbp-0x10]
20: 48 8b 45 f8          mov   rax,QWORD PTR [rbp-0x8]
24: 48 89 ce             mov   rsi,rcx
27: 48 89 c7             mov   rdi,rax
2a: e8 00 00 00 00       call 2f <searchorig+0x2f>
2f: c9                   leave
30: c3                   ret

0000000000000031 <linearSearch>:
31:  f3 0f 1e fa          endbr64
35:  55                   push  rbp
36:  48 89 e5             mov   rbp,rsi
39: 48 89 7d e8          mov   QWORD PTR [rbp-0x18],rdi
3d: 48 89 75 e0          mov   QWORD PTR [rbp-0x20],rsi
41: 48 89 55 d8          mov   QWORD PTR [rbp-0x28],rdx
45: 48 c7 45 f8 00 00 00 mov   QWORD PTR [rbp-0x8],0x0
4c: 00
4d: eb 2a               jmp   79 <linearSearch+0x48>
4f: 48 8b 45 f8          mov   rax,QWORD PTR [rbp-0x8]
53: 48 8d 14 c5 00 00 00 lea   rdx,[rax*8+0x0]
5a: 00
5b: 48 8b 45 e8          mov   rax,QWORD PTR [rbp-0x18]
5f: 48 01 d0             add   rax,rdx
62: 48 8b 10             mov   rdx,QWORD PTR [rax]
65: 48 8b 45 d8          mov   rax,QWORD PTR [rbp-0x28]
69: 48 39 c2             cmp   rdx,rax
6c: 75 06               jne   74 <linearSearch+0x43>
6e: 48 8b 45 f8          mov   rax,QWORD PTR [rbp-0x8]
```

### 1.3. Preparatory Steps: Pintool

1. Change directory into **assignment1/tracerorig** folder

```
ssh-prashantnair@ssh.ece.ubc.ca — ssh prashantnair@ssh.ece.ubc.ca — 130x34
ssh-prashantnair@ssh.ece.ubc.ca ~
ssh-linux4:~/assignment1/algorithms> cd ../tracerorig/
ssh-linux4:~/assignment1/tracerorig> ls
clean_tracer.sh  makefile  makefile.rules  make_tracer.sh  obj-intel64  quick_make.sh  README  tracer.cpp
```

2. Understand the **assignment1/tracerorig** folder

```
ssh-prashantnair@ssh.ece.ubc.ca — ssh prashantnair@ssh.ece.ubc.ca — 130x34
ssh-prashantnair@ssh.ece.ubc.ca ~
ssh-linux4:~/assignment1/algorithms> cd ../tracerorig/
ssh-linux4:~/assignment1/tracerorig> ls
clean_tracer.sh  makefile  makefile.rules  make_tracer.sh  obj-intel64  quick_make.sh  README  tracer.cpp
ssh-linux4:~/assignment1/tracerorig>
```

3. Understand the **tracer.cpp** file and compile your code

```
ssh-prashantnair@ssh.ece.ubc.ca — ssh prashantnair@ssh.ece.ubc.ca — 130x34
ssh-prashantnair@ssh.ece.ubc.ca ~
ssh-linux4:~/assignment1/tracerorig> vim tracer.cpp
```

```
37 /* ===== */
38 // Global variables
39 /* ===== */
40
41 FILE* out;
42 bool output_file_closed = false;
43
44 /* ===== */
45 // Add your variables below this
46 /* ===== */
47
48
49
50
51
52
53
54
55
56 /* ===== */
57 // Add your variables above this
58 /* ===== */
```

```
113 /* ===== */
114 // Instrumentation callbacks
115 /* ===== */
116
117 // Is called for every instruction
118 VOID Instruction(INS ins, VOID *v)
119 {
120     // begin each instruction with this function
121     UINT32 opcode = INS_Opcode(ins);
122     INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)BeginInstruction, IARG_INST_PTR, IARG_UINT32, opcode, IARG_END);
123     // ===== ADD YOUR CODE BELOW THIS POINT =====
124
125
126
127
128
129
130
131
132
133
134     // ===== ADD YOUR CODE ABOVE THIS POINT =====
135     // finalize each instruction with this function
136     INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)EndInstruction, IARG_END);
137 }
138
139 VOID Fini(INT32 code, VOID *v)
140 {
141     assert(instrCount == (nonmeminstCount+meminstCount));
142     fprintf(out, "instrCount,%ld\n", instrCount);
143     fprintf(out, "nonmeminstCount,%ld\n", nonmeminstCount);
144     fprintf(out, "meminstCount,%ld\n", meminstCount);
145 }
```

```

80 }
81
82
83 /* ===== */
84 // Analysis routines
85 /* ===== */
86
87 void BeginInstruction(VOID *lp, UINT32 op_code, VOID *opstring){
88     instrCount++;
89     if(instrCount%1000000 == 0){
90         std::cout << " " << std::endl;
91         if(instrCount%1000000 == 0){
92             std::cout << " -- " << instrCount/1000000 << " Million Instructions" << std::endl << std::flush;
93         }
94     }
95 }
96
97 /* ===== ADD YOUR FUNCTION (IF ANY) BELOW ===== */
98
99
100
101
102
103
104
105
106 /* ===== ADD YOUR FUNCTION (IF ANY) ABOVE ===== */
107
108
109 void EndInstruction()
110 {
111
112 }

```

```

ssh-linux4:~/assignment1/tracerorig$ bash quick_make.sh
g++ -Wall -Werror -Wno-unknown-pragmas -D__PIN__=1 -DPIN_CRT=1 -fno-stack-protector -fno-exceptions -funwind-tables -fasynchronous-unwind-tables -fno-rtti -DTARGET_IA32E -DHOST_IA32E -fPIC -DTARGET_LINUX -fabi-version=2 -faligned-new -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//source/include/pin -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//source/include/pin/gen -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/stlport/include -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/libstdc++/include -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include/arch-x86_64 -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include/kernel/uapi -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include/kernel/uapi/asm-x86 -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/components/include -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/xed-intel64/include/xed -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//source/tools/Utils -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//source/tools/InstLib -O3 -fomit-frame-pointer -fno-strict-aliasing -c -o obj-intel64/tracer.o tracer.cpp
g++ -shared -Wl,--hash-style=sysv /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//intel64/runtime/pincrt/crtbeginS.o -Wl,-Bsymbolic -Wl,--version-script=/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//source/include/pin/pintool.ver -fabi-version=2 -o obj-intel64/tracer.so obj-intel64/tracer.o -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//intel64/runtime/pincrt -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//intel64/lib -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//intel64/lib-ext -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/xed-intel64/lib -lpin -lxd /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir//intel64/runtime/pincrt/crtendS.o -lpin3dwarf -ldl -dynamic -nostdlib -lstlport-dynamic -lm -dynamic -lc -dynamic -lunwind-dynamic
ssh-linux4:~/assignment1/tracerorig$

```

4. Refer to Pintool [1] website to write/understand/explore some example Pintools. You have a folder called **examples**, implement Pintools there and compile them.

```

ssh-linux4:~/assignment1/tracerorig$ cd ../example/
ssh-linux4:~/assignment1/example$ ls
clean_example.sh example.cpp make_example.sh makefile makefile.rules obj-intel64 pintool.log quick_make.sh README
ssh-linux4:~/assignment1/example$

```

5. Execute your Pintool to “instrument” **assignment1/algorithms/assignment1.bin**. Note that the first parameter after assignment1.bin is “0” → Indicating Linear-Search

```

jhw0036@gattaca:~/ta/cpen411/2023/assignment1_test/tracerorig$ ../pin-dir/pin -t obj-intel64/tracer.so -m 0 -- ../algorithms/assignment1.bin 0
..... -- 10 Million Instructions
..... -- 20 Million Instructions
..... -- 30 Million Instructions
..... -- 40 Million Instructions
..... -- 50 Million Instructions
..... -- 60 Million Instructions
..... -- 70 Million Instructions
..... -- 80 Million Instructions
..... -- 90 Million Instructions
..... -- 100 Million Instructions
..... -- 110 Million Instructions

```

```

ssh-linux4:~/assignment1/tracerorig$ cd ../output/
ssh-linux4:~/assignment1/output$ vim tracerorig.out

```



```

instrCount,115515776
nonmeminstCount,0
meminstCount,0
branchCount,0
MEMDELAY,2000
NONMEMDELAT,1000
CYCLEVAL-SERIAL,0
CYCLEVAL-HIDEMEM,0
CHANGE,0
OPT-MEMDELAY,2000
OPT-NONMEMDELAT,1000
OPT-CYCLEVAL-SERIAL,0
OPT-CYCLEVAL-HIDEMEM,0
SPEEDUP,nan
SPEEDUP-HIDEMEM,nan

```

6. Add a constant delay = NONMEMDELAY to the CYCLEVALS counter after each instruction. In this example, we do not know which instructions are memory instructions and we assume every instruction to be a non-memory instruction.

```

prashantnair — ssh prashantnair@ssh.ece.ubc.ca — 130x34
ssh prashantnair@ssh.ece.ubc.ca ~/Downloads — -bash

INT32 Usage()
{
    cerr << "This tool creates helps instrument instructions" << endl << "Percentage Change in Non-Memory Instruction Latency:
From -90 to +90 with -o" << endl;
    cerr << KNOB_BASE::StringKnobSummary() << endl;
    return -1;
}

/* ===== */
// Analysis routines
/* ===== */

void BeginInstruction(VOID *ip, UINT32 op_code, VOID *opstring){
    instrCount++;
    //This is an example, and this is an incorrect assumption
    CYCLEVALS = CYCLEVALS + NONMEMDELAY;

    if(instrCount%1000000 == 0){
        std::cout << "." << std::flush;
        if(instrCount%10000000 == 0){
            std::cout << " -- " << instrCount/1000000 << " Million Instructions" << std::endl << std::flush;
        }
    }
}

/* ++++++ ADD YOUR FUNCTION (IF ANY) BELOW ++++++ */

```

```

prashantnair — ssh prashantnair@ssh.ece.ubc.ca — 130x34
ssh prashantnair@ssh.ece.ubc.ca ~/Downloads — -bash

ssh-linux4:~/assignment1/tracerorig> bash quick_make.sh
g++ -Wall -Werror -Wno-unknown-pragmas -D__PIN__=1 -DPIN_CRT=1 -fno-stack-protector -fno-exceptions -funwind-tables -fasynchronous-unwind-tables -fno-rtti -DTARGET_IA32E -DHOST_IA32E -fPIC -DTARGET_LINUX -fabi-version=2 -faligned-new -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/source/include/pin -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/source/include/pin/gen -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/stlport/include -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/libstdc++/include -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include/arch-x86_64 -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include/kernel/uapi -isystem /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/crt/include/kernel/uapi/asm-x86 -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/components/include -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/xed-intel64/include/xed -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/source/tools/Utils -I/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/source/tools/InstLib -O3 -fomit-frame-pointer -fno-strict-aliasing -c -o obj-intel64/tracer.o tracer.cpp
g++ -shared -Wl,--hash-style=sysv /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/intel64/runtime/pincrt/crtbegin.S.o -Wl,-Bsymbolic -Wl,--version-script=/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/source/include/pin/pintool.ver -fabi-version=2 -o obj-intel64/tracer.so obj-intel64/tracer.o -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/intel64/runtime/pincrt -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/intel64/lib -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/intel64/lib-ext -L/ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/extras/xed-intel64/lib -lpin -lxed /ubc/ece/home/pnj/faculty/prashantnair/assignment1/pin-dir/intel64/runtime/pincrt/crtend.S.o -lpin3dwarf -ldl -dynamic -nostdlib -lstdport-dynamic -lm-dynamic -lc-dynamic -lunwind-dynamic
ssh-linux4:~/assignment1/tracerorig>

```



```

jhw0036@gattaca:~/ta/cpen411/2023/assignment1_test/tracerorig$ ../pin-dir/pin -t obj-intel64/tracer.so -m 0 -- ./algorithms/assignment1.bin 0
..... -- 10 Million Instructions
..... -- 20 Million Instructions
..... -- 30 Million Instructions
..... -- 40 Million Instructions
..... -- 50 Million Instructions
..... -- 60 Million Instructions
..... -- 70 Million Instructions
..... -- 80 Million Instructions
..... -- 90 Million Instructions
..... -- 100 Million Instructions
..... -- 110 Million Instructions

```

Your Student #

```

prashantnair — ssh prashantnair@ssh.ece.ubc.ca — 130x34
~ — ssh prashantnair@ssh.ece.ubc.ca
ssh-linux4:~/assignment1/tracerorig> cd ../output/
ssh-linux4:~/assignment1/output> vim tracerorig.out

```

```

instrCount,115515776
nonmeminstCount,0
meminstCount,0
branchCount,0
MEMDELAY,2000
NONMEMDELAT,1000
CYCLEVAL-SERIAL,115515776000
CYCLEVAL-HIDEMEM,0
CHANGE,0
OPT-MEMDELAY,2000
OPT-NONMEMDELAT,1000
OPT-CYCLEVAL-SERIAL,0
OPT-CYCLEVAL-HIDEMEM,0
SPEEDUP,inf
SPEEDUP-HIDEMEM,nan

```

7. Remove any modification to tracer.c and proceed to the next step. Essentially remove the line `CYCLEVALS = CYCLEVALS + NONMEMDELAY`.

**Note:** Your Pintool should not execute longer than 5 mins on the server. If it takes longer to execute, we will give you 0 marks. Please write efficient programs and Pintools.

## Chapter 2

# Evaluation: Algorithm & Pintool [5 Points]

1. **[0.5 Points]** Write a Pintool in `assignment1/traceorig/` to count the number of instructions in `assignment1/algorithms/assignment1.bin` for Linear Search (This should already be implemented in the preparatory example).
2. **[1 Point]** Add a function in this Pintool to count the number of memory and non-memory instructions in `assignment1/algorithms/assignment1.bin` for Linear Search.
  - “**meminstCount**” and “**nonmeminstCount**” variables must be repeatedly updated.
  - **Note:** There is an assertion and if `(meminstCount + nonmeminstCount) != instrCount`, your program will fail. It is non-trivial to count memory instructions. You will need to understand the functions `INS_IsMemoryRead()` and `INS_IsMemoryWrite()` for this. These functions are described here: [Memory Read and Write Functions](#).
  - Additionally, please refer to an example here: [Memory Traces](#).
3. **[0.5 Points]** Add a function in this Pintool to count the number of **branch instructions** in `assignment1/algorithms/assignment1.bin` for Linear Search.
  - The “**branchCount**” variable needs to be updated in this example.
  - **Note:** You must understand the functions `INS_IsBranch()` for this. These functions are described here: [Branch Count Function](#).
4. **[0.5 Points]** Assuming all instructions are executed one after another with their own delays, compute the **total execution time** required to execute `assignment1/algorithms/assignment1.bin` for Linear Search. For this exercise, the Pintool is pre-filled with delays (CPI) for memory and nonmemory instructions.
  - Your **CYCLEVALS** variable needs to be repeatedly updated in this example. This needs to be done by using the values of **MEMDELAY** for memory instructions and **NONMEMDELAY** for non-memory instructions.
  - The output file, `tracerorig.out` will reflect this number.
5. **[0.5 Points]** Assuming all memory instructions do not show any latency or block any other instructions (perhaps due to some computer architecture optimization), compute the **total execution time** required to execute `assignment1/algorithms/assignment1.bin` for Linear Search. For this exercise, the Pintool is pre-filled with delays (CPI) for memory and non-memory instructions. **Note:** Your **CYCLEVALP** variable needs to be repeatedly updated in this example.
6. **[0.5 Points]** You can change the latency (CPI) of non-memory instructions by +90% to -90%. Repeat steps 4 and 5. On canvas, enter this value (such as 10%, 20%, -50%, etc.) wherein the speedup as compared to step 4 **just crosses 1.2. This value must be entered as a comment on Canvas.** This has trade-offs, a 50% increase in non-memory instruction CPI causes a 50% decrease in memory instruction CPI.
  - Your **OPTCYCLEVALP** and **OPTCYCLEVALS** variables need to be repeatedly updated in this example. This needs to be done by using the values of **MEMDELAY** for memory instructions and **NONMEMDELAY** for non-memory instructions.

```
ssh-linux4:~/assignment1/tracerorig> ../pin-dir/pin -t obj-intel64/tracer.so -m -20 -- ../algorithms/assignment1.bin 0 Your Student #
```

- The input parameter to your pintool “-m” is used to pass the change in latency.
- Remember, you need to report the value used when your SPEEDUP in your output file just surpasses 1.2.
- This value needs to reflect in the output file **../output/traceorig.out**. The entry “**CHANGE**” should show this value. Make sure you do not change this value after you converge on the right value. If you change “-m” and run your experiment again, your “**CHANGE**” value in the output folder would be overwritten.

7. **[0.5 Points]** Change the directory into assignment1/algorithms folder and implement your searching algorithm – in **searchnew.c** and **searchnew.h**.

- You can only edit these files, and you can add any number of additional functions in **searchnew.h** and **searchnew.c**
- This algorithm needs to **PASS**. After you execute bash runall.sh in this folder, check the **../output/searchnew.txt**

8. **[0.5 Points]** Change directory into assignment1/tracernew folder. Write a Pintool in its trace.cpp to count the number of instructions in assignment1/algorithms/assignment1.bin for <your searching algorithm>.

```
ssh-linux4:~/assignment1> cd tracernew/
ssh-linux4:~/assignment1/tracernew> vim tracer.cpp
ssh-linux4:~/assignment1/tracernew> ../pin-dir/pin -t obj-intel64/tracer.so -m 0 -- ../algorithms/assignment1.bin 1 Your Student #
```

9. **[0.5 Points]** Ensure that <your searching algorithm> executes at least **20x lower total instructions** as compared to your original Linear-Search. You can check the output folder/tracernew.out file for the total instructions executed.

# Chapter 3

## Submission Instructions

1. To submit, please execute the following command within assignment1 folder.

```
jhw0036@ssh-linux3:~/assignment1$ ls
algorithms  create_submitarchive.sh  example  output  pin-dir  tracernew  tracerorig
jhw0036@ssh-linux3:~/assignment1$ ./create_submitarchive.sh
algorithms/
algorithms/assignment1.o
algorithms/searchnew.h
algorithms/searchoriginal.h
algorithms/searchoriginal.c
algorithms/Makefile
algorithms/README
algorithms/searchnew.c
algorithms/runall.sh
output/
tracernew/
tracernew/tracer.cpp
tracernew/clean_tracer.sh
tracernew/quick_make.sh
tracernew/obj-intel64/
tracernew/obj-intel64/tracer.so
tracernew/obj-intel64/tracer.o
tracernew/makefile
tracernew/README
tracernew/make_tracer.sh
tracernew/makefile.rules
tracerorig/
tracerorig/tracer.cpp
tracerorig/clean_tracer.sh
tracerorig/quick_make.sh
tracerorig/obj-intel64/
tracerorig/obj-intel64/tracer.so
tracerorig/obj-intel64/tracer.o
tracerorig/makefile
tracerorig/README
tracerorig/make_tracer.sh
tracerorig/makefile.rules
jhw0036@ssh-linux3:~/assignment1$ ls
algorithms  create_submitarchive.sh  example  output  pin-dir  submission.tar.gz  tracernew  tracerorig
jhw0036@ssh-linux3:~/assignment1$
```

2. You can then run **rsync** from your local computer and download the submission file.

```
MacBook-Pro-10:Assignment-1 prashantnair@ssh.ece.ubc.ca$ rsync -rtv prashantnair@ssh.ece.ubc.ca:~/assignment1/submission.tar.gz .
receiving incremental file list
submission.tar.gz

sent 43 bytes  received 1,848,394 bytes  1,232,291.33 bytes/sec
total size is 1,847,849  speedup is 1.00
```

3. Please upload **ONLY** the **submission.tar.gz** on Canvas. Make sure it has the updated algorithms, tracer.cpp, and output files. **If we find stale files and execute those, you can get “0” points for this assignment.**

Thus, be extremely careful and double check if your outputs, source files, etc. are the right ones! We have 80 students in this course, and we will not re-evaluate your Assignment if you submit incorrect or stale work.

# Bibliography

- [1] Intel Inc. Pintool Software. <https://software.intel.com/sites/landingpage/pintool/docs/98484/Pin/html/index.html>.
- [2] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," *SIGPLAN Not.*, vol. 40, no. 6, p. 190–200, jun 2005. [Online]. Available: <https://doi.org/10.1145/1064978.1065034>
- [3] Homebrew. Website Access for Homebrew. <https://brew.sh/>.
- [4] Vscod. Website Access for Remote Development using SSH - Visual Studio Code. <https://code.visualstudio.com/docs/remote/ssh>.
- [5] Cyberduck. Website Access for Cyberduck. <https://cyberduck.io/>.
- [6] MobaXterm. Website Access for MobaXterm . <https://mobaxterm.mobatek.net/>.