

COMP30640: Operating Systems – Finbar Allan

Assignment 2

Question 1

Student ID: 22208108
Initial Value: 2000000
Expected Solution: 24208108

The adder.s program has been modified to add the value of my student number to the value of the two numbers summed in the original program by adding the following operation to load the value of my student number into the a2 register:

```
li a2,22208108
```

The next modification required is to sum my student number with the value of the two numbers summed in the original program using the add operation. The target of this operation will be the a0 register, the location in which the original value was registered. We can overwrite this value to ensure the desired value is printed from the a0 register with the following operation:

```
add a0,a0,a2
```

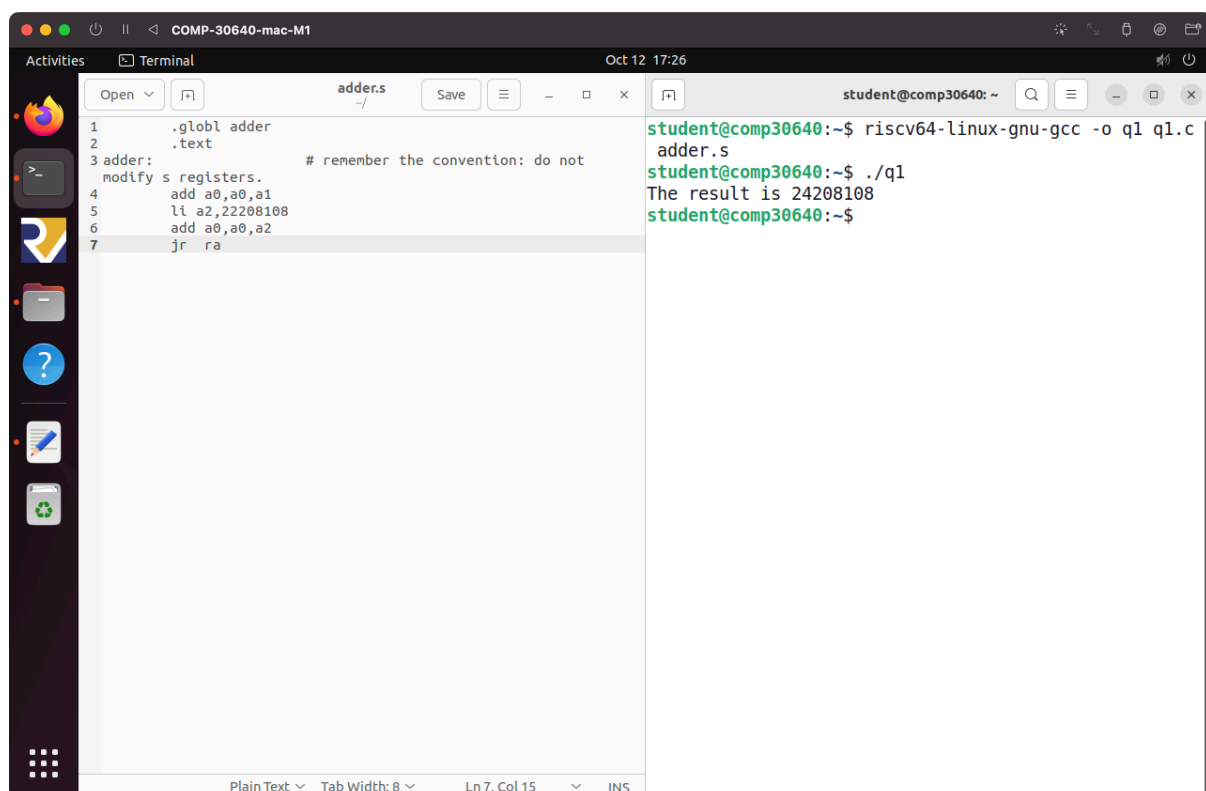


Figure 1: Modified adder.s program in text editor with terminal output from compiling and executing the file displayed side-by-side.

The following code block shows the fully modified adder.s program which has also been attached as a separate file as part of this assignment submission:

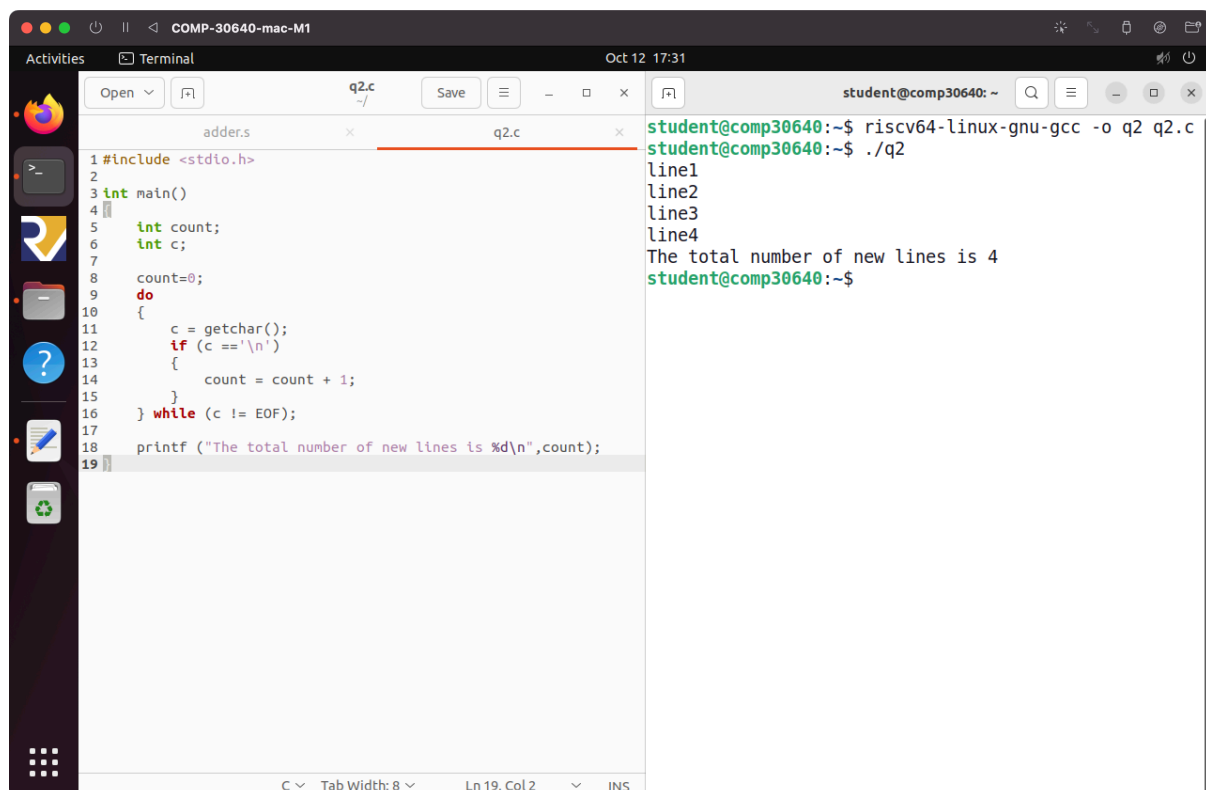
```
.globl adder
.text
adder:                # remember the convention: do not modify s
registers.
    add a0,a0,a1
    li a2,22208108
    add a0,a0,a2
    jr  ra
```

Question 2

The purpose of the q2.c program was modified to count the number of newline characters in a user input from the original program to count the total number of characters in a user input by updating the conditional if statement to the following line:

```
if (c=='\n')
```

This modification allows the do while loop to check if each character the getChar() function reads from the user input is a newline character and increments the counter when this condition evaluates to true as is displayed in Figure 2:



The screenshot shows a text editor window with the following code in q2.c:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int count;
6     int c;
7
8     count=0;
9     do
10    {
11        c = getchar();
12        if (c == '\n')
13        {
14            count = count + 1;
15        }
16    } while (c != EOF);
17
18    printf("The total number of new lines is %d\n",count);
19 }
```

The terminal window shows the following output:

```
student@comp30640:~$ riscv64-linux-gnu-gcc -o q2 q2.c
student@comp30640:~$ ./q2
line1
line2
line3
line4
The total number of new lines is 4
student@comp30640:~$
```

Figure 2: Modified q2.c program in text editor with terminal output from compiling, testing, and executing the file displayed side-by-side.

The fully modified q2.c program has been attached as a separate file as part of this assignment submission and is also displayed in plain text below:

```
#include <stdio.h>

int main()
{
    int count;
    int c;
    count=0;
    do
    {
        c = getchar();
        if (c == '\n')
        {
            count = count + 1;
        }
    } while (c != EOF);
    printf ("The total number of new lines is %d\n",count);
}
```

Question 3

The main.c program has been updated to print my student number with the addition of another print statement after the OS greeting as displayed below:

```
printf("xv6 kernel is booting\n");
printf("My Student ID: 22208108\n");
```

The xv6 rebuild and execution is carried out with the following commands before being terminated as is displayed in Figure 3:

```
cd /home/student/xv6-riscv
make qemu
```

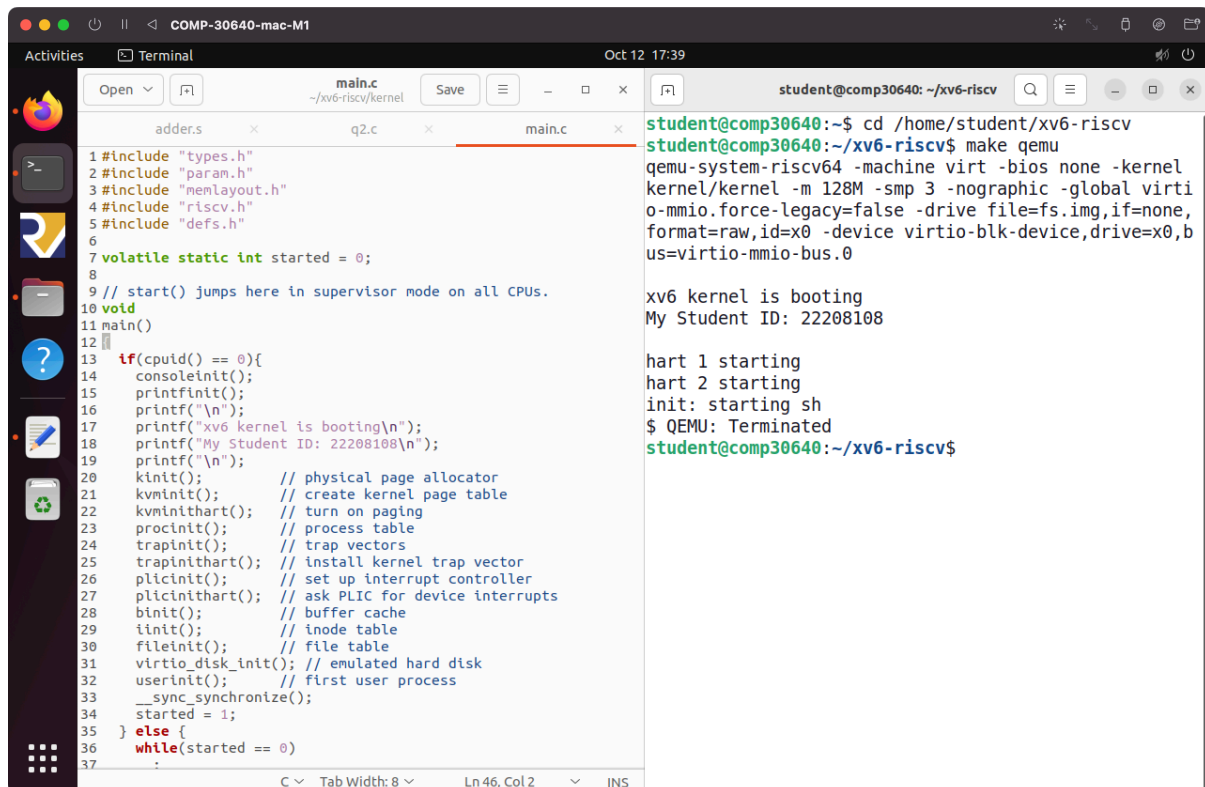


Figure 3: Modified main.c program in text editor with terminal output from building and running XV6 operating system commands displayed side-by-side.

As part of this assignment the modified main.c file has been attached in the submission file with the full program being displayed below:

```

#include "types.h"
#include "param.h"
#include "memlayout.h"
#include "riscv.h"
#include "defs.h"

```

```
volatile static int started = 0;
```

```
// start() jumps here in supervisor mode on all CPUs.
```

```
void
```

```
main()
```

```
{
```

```
    if(cpuuid() == 0){
```

```
        consoleinit();
```

```
        printfinit();
```

```
        printf("\n");
```

```
        printf("xv6 kernel is booting\n");
```

```
        printf("My Student ID: 22208108\n");
```

```
        printf("\n");
```

```
        kinit();           // physical page allocator
```

```
        kvminit();         // create kernel page table
```

```
        kvminithart();     // turn on paging
```

```
        procinit();        // process table
```

```
        trapinit();        // trap vectors
```

```
        trapinithart();    // install kernel trap vector
```

```

    plicinit();          // set up interrupt controller
    plicinithart();      // ask PLIC for device interrupts
    binit();             // buffer cache
    iinit();             // inode table
    fileinit();          // file table
    virtio_disk_init();  // emulated hard disk
    userinit();          // first user process
    __sync_synchronize();
    started = 1;
} else {
    while(started == 0)
        ;
    __sync_synchronize();
    printf("hart %d starting\n", cpuid());
    kvminithart();       // turn on paging
    trapinithart();      // install kernel trap vector
    plicinithart();      // ask PLIC for device interrupts
}

scheduler();
}

```