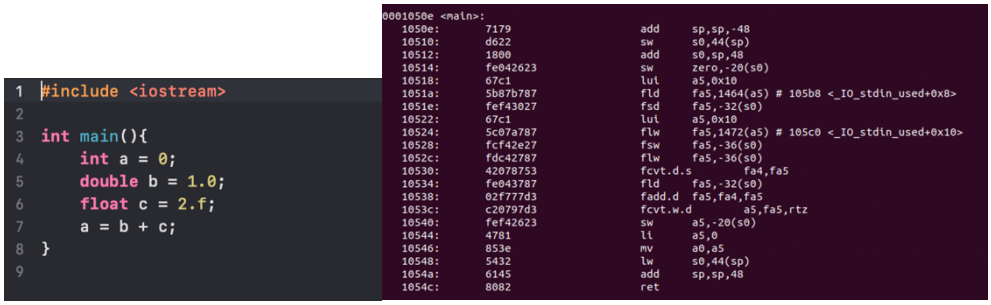
# Computer Organization Project 5 – GNU Toolchain and RISC-V ISA

B11015059 賴均

- Write a report to elaborate the compiled RISC-V assembly code in correspondence of your C/C++ source code.

1. Jyun01.cpp & Jyun01.asm



```
#include <iostream>

int main(){
    int a = 0;
    double b = 1.0;
    float c = 2.f;
    a = b + c;
}

```

```
0001050e <main>:
   1050e:       7179                add     sp,sp,-48
   10510:       d622                sw      s0,44(sp)
   10512:       1800                add     s0,sp,48
   10514:       fe042623            sw      zero,-20(s0)
   10518:       67c1                lui     a5,0x10
   1051a:       5b87b787            fld     fa5,1464(a5) # 105b8 <_IO_stdin_used+0x8>
   1051e:       fef43027            fsd     fa5,-32(s0)
   10522:       67c1                lui     a5,0x10
   10524:       5c07a787            flw     fa5,1472(a5) # 105c0 <_IO_stdin_used+0x10>
   10528:       fcf42e27            fsw     fa5,-36(s0)
   1052c:       fdc42787            flw     fa5,-36(s0)
   10530:       42078753            fcvt.d.s        fa4,fa5
   10534:       fe043787            fld     fa5,-32(s0)
   10538:       02f777d3            fadd.d  fa5,fa4,fa5
   1053c:       c20797d3            fcvt.w.d        a5,fa5,rtz
   10540:       fef42623            sw      a5,-20(s0)
   10544:       4781                li      a5,0
   10546:       853e                mv      a0,a5
   10548:       5432                lw      s0,44(sp)
   1054a:       6145                add     sp,sp,48
   1054c:       8082                ret
```

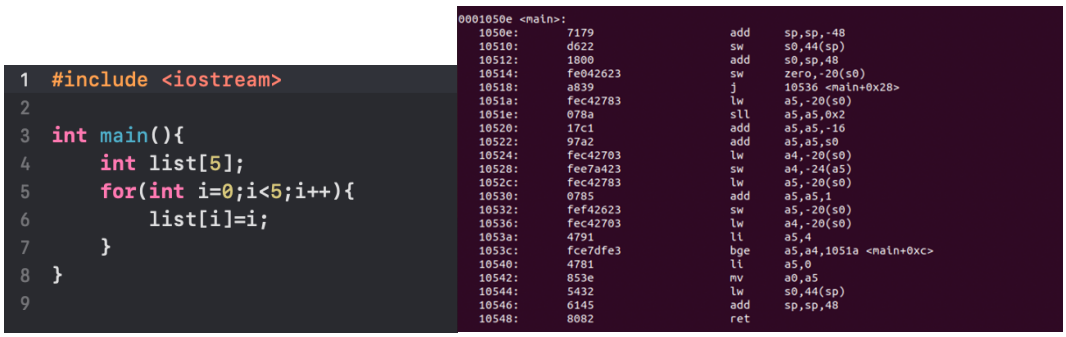
sw zero,-20(s0)　　設定變數 a

fld fa5,1464(a5)　　設定變數 b

flw fa5,1472(a5)　　設定變數 c

fadd.d fa5,fa4,fa5　a = b + c

2. Jyun02.cpp & Jyun02.asm



```
#include <iostream>

int main(){
    int list[5];
    for(int i=0;i<5;i++){
        list[i]=i;
    }
}

```

```
0001050e <main>:
   1050e:       7179                add     sp,sp,-48
   10510:       d622                sw      s0,44(sp)
   10512:       1800                add     s0,sp,48
   10514:       fe042623            sw      zero,-20(s0)
   10518:       a839                j       10536 <main+0x28>
   1051a:       fec42783            lw      a5,-20(s0)
   1051e:       078a                sll     a5,a5,0x2
   10520:       17c1                add     a5,a5,-16
   10522:       97a2                add     a5,a5,s0
   10524:       fec42703            lw      a4,-20(s0)
   10528:       fee7a423            sw      a4,-24(a5)
   1052c:       fec42783            lw      a5,-20(s0)
   10530:       0785                add     a5,a5,1
   10532:       fef42623            sw      a5,-20(s0)
   10536:       fec42703            lw      a4,-20(s0)
   1053a:       4791                li      a5,4
   1053c:       fce7dfe3            bge     a5,a4,1051a <main+0xc>
   10540:       4781                li      a5,0
   10542:       853e                mv      a0,a5
   10544:       5432                lw      s0,44(sp)
   10546:       6145                add     sp,sp,48
   10548:       8082                ret
```

add a5,a5,1　　i++

bge a5,a4,1051a <main+0xc>　　判斷 i 是否小於 5

3. Jyun03.cpp & Jyun03.asm

```cpp
#include <iostream>

int a = 10;
int main(){
    int b = 20;
    a = a + b;
}

```

```
0001050e <main>:
   1050e:       1101                    add     sp,sp,-32
   10510:       ce22                    sw      s0,28(sp)
   10512:       1000                    add     s0,sp,32
   10514:       47d1                    li      a5,20
   10516:       fef42623                sw      a5,-20(s0)
   1051a:       8201a703                lw      a4,-2016(gp) # 12020 <a>
   1051e:       fec42783                lw      a5,-20(s0)
   10522:       973e                    add     a4,a4,a5
   10524:       82e1a023                sw      a4,-2016(gp) # 12020 <a>
   10528:       4781                    li      a5,0
   1052a:       853e                    mv      a0,a5
   1052c:       4472                    lw      s0,28(sp)
   1052e:       6105                    add     sp,sp,32
   10530:       8082                    ret
```

lw a4,-2016(gp) # 12020 <a>　　將 a = 10 讀出來

lw a5,-20(s0)　　設定 b=20

add a4,a4,a5　　a = a + b

sw a4,-2016(gp) # 12020 <a>　　將 a 的結果放回去

- **Compare both the elf-gcc and linux-gnu-gcc compilation results with and without using the -static compilation option by using objdump.**

  觀察後發現

  1. linux-gnu-gcc
     使用 linux-gnu-gcc 時，最後的 assembly static 會比沒有 static 多好幾萬行。
  2. elf-gcc
     在使用 elf-gcc 時，則沒有明顯的差異。

- **In your program, declare a variety of C/C++ variable types with and without non-zero initialization and identify the actual physical locations in either final binary program or run-time memory. Summarize your observations.**

  global variable 在程式運行期間會一直存在於固定的記憶體位置，無論在程式的哪個地方都可以訪問到它們。當程式執行結束時，全域變數的記憶體位置才會被釋放回去，如果全域變數沒有被初始化，它們的預設值會是 0。

  local variable 在被宣告時會在 stack pointer 上動態分配一段記憶體空間，並且存儲初始值。如果區域變數沒有初始值，則它們的值會是該記憶體位置上原本存在的值。區域變數的生命週期在其所屬的區塊結束時就會結束，該記憶體空間會被歸還回去。因此，每次執行時，區域變數的記憶體位置都不是固定的，會隨著程式執行的流程而改變。