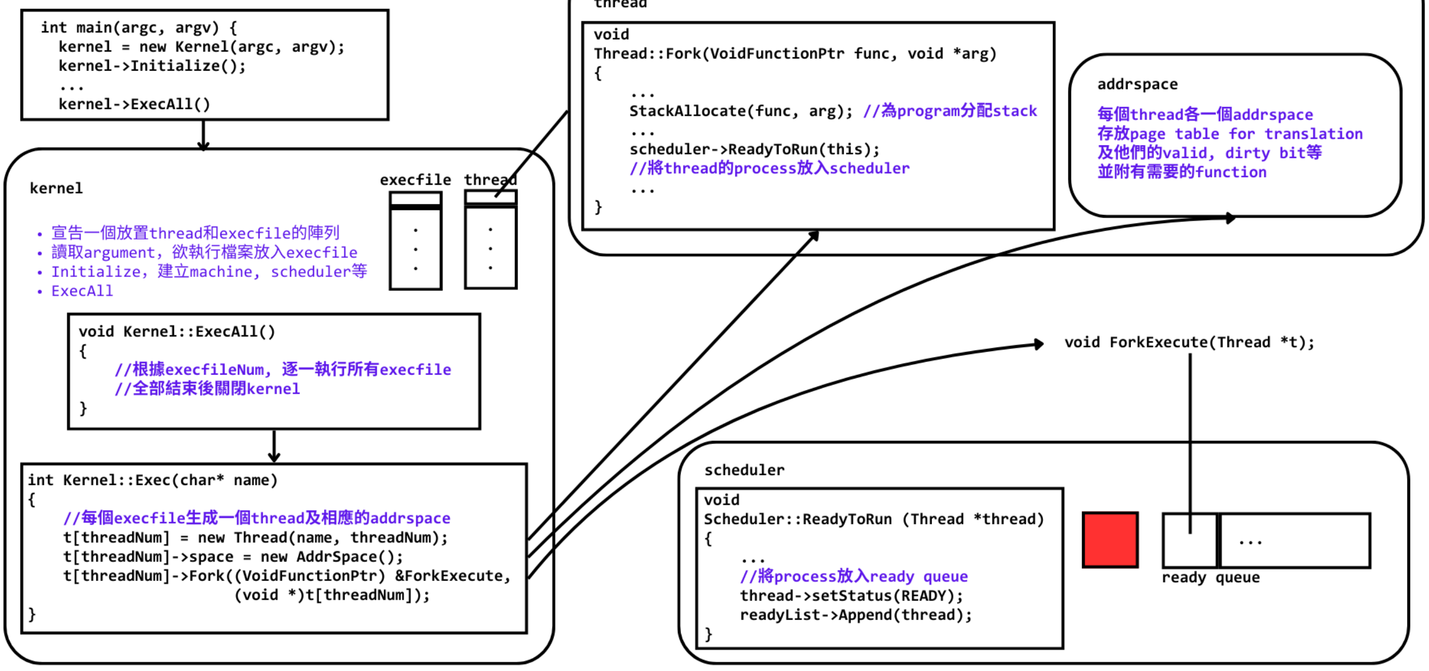
110705017 何翊華 110705063 廖偉辰

Part 1:Trace Code

1. **Explain function**
    1. **threads/thread.cc**
    2. **userprog/addrspace.cc**
    3. **threads/kernel.cc**
    4. **threads/scheduler.cc**

Generate



```
int main(argc, argv) {
  kernel = new Kernel(argc, argv);
  kernel->Initialize();
  ...
  kernel->ExecAll()
```

kernel

- 宣告一個放置thread和execfile的陣列
- 讀取argument，欲執行檔案放入execfile
- Initialize，建立machine，scheduler等
- ExecAll

```
void Kernel::ExecAll()
{
    //根據execfileNum，逐一執行所有execfile
    //全部結束後關閉kernel
}
```

```
int Kernel::Exec(char* name)
{
    //每個execfile生成一個thread及相應的addrspace
    t[threadNum] = new Thread(name, threadNum);
    t[threadNum]->space = new AddrSpace();
    t[threadNum]->Fork((VoidFunctionPtr) &ForkExecute,
                       (void *)t[threadNum]);
}
```

thread

```
void
Thread::Fork(VoidFunctionPtr func, void *arg)
{
    ...
    StackAllocate(func, arg); //為program分配stack
    ...
    scheduler->ReadyToRun(this);
    //將thread的process放入scheduler
    ...
}
```

addrspace

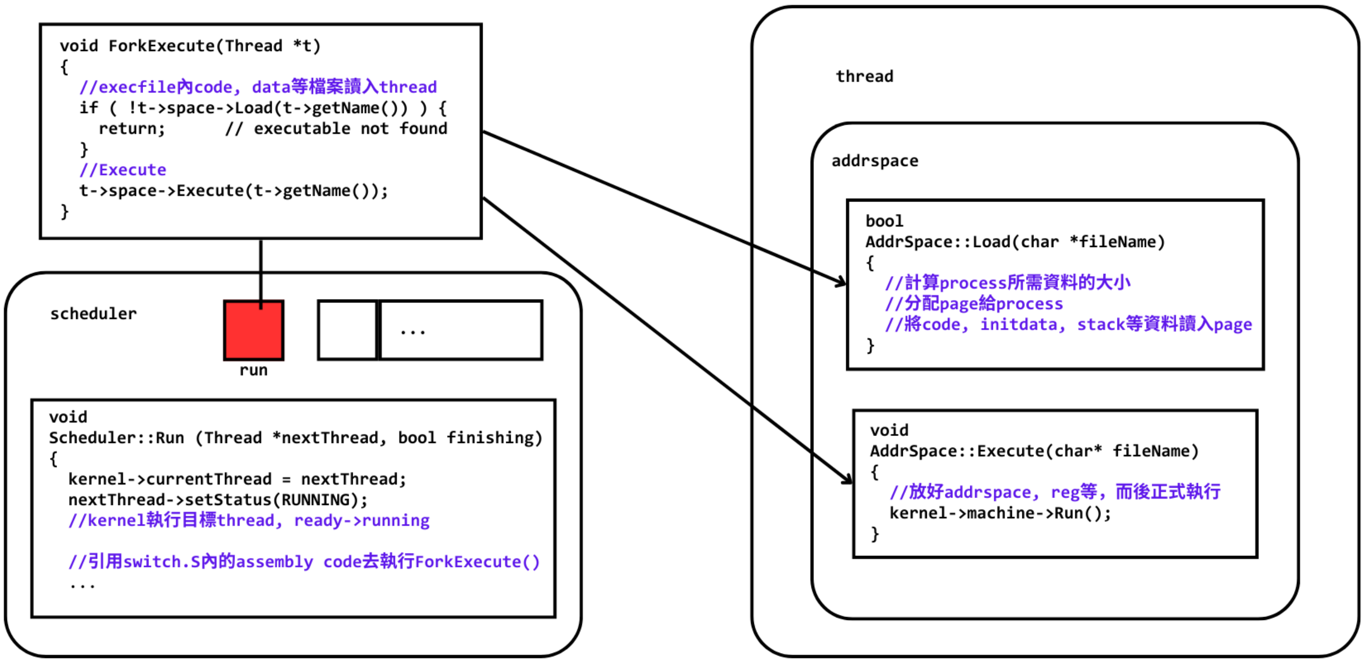
每個thread各一個addrspace
存放page table for translation
及他們的valid, dirty bit等
並附有需要的function

```
void ForkExecute(Thread *t);
```

scheduler

```
void
Scheduler::ReadyToRun (Thread *thread)
{
    ...
    //將process放入ready queue
    thread->setStatus(READY);
    readyList->Append(thread);
}
```

run



```
void ForkExecute(Thread *t)
{
  //execfile內code, data等檔案讀入thread
  if ( !t->space->Load(t->getName()) ) {
    return;       // executable not found
  }
  //Execute
  t->space->Execute(t->getName());
}
```

scheduler

```
void
Scheduler::Run (Thread *nextThread, bool finishing)
{
  kernel->currentThread = nextThread;
  nextThread->setStatus(RUNNING);
  //kernel執行目標thread, ready->running

  //引用switch.S內的assembly code去執行ForkExecute()
  ...
```

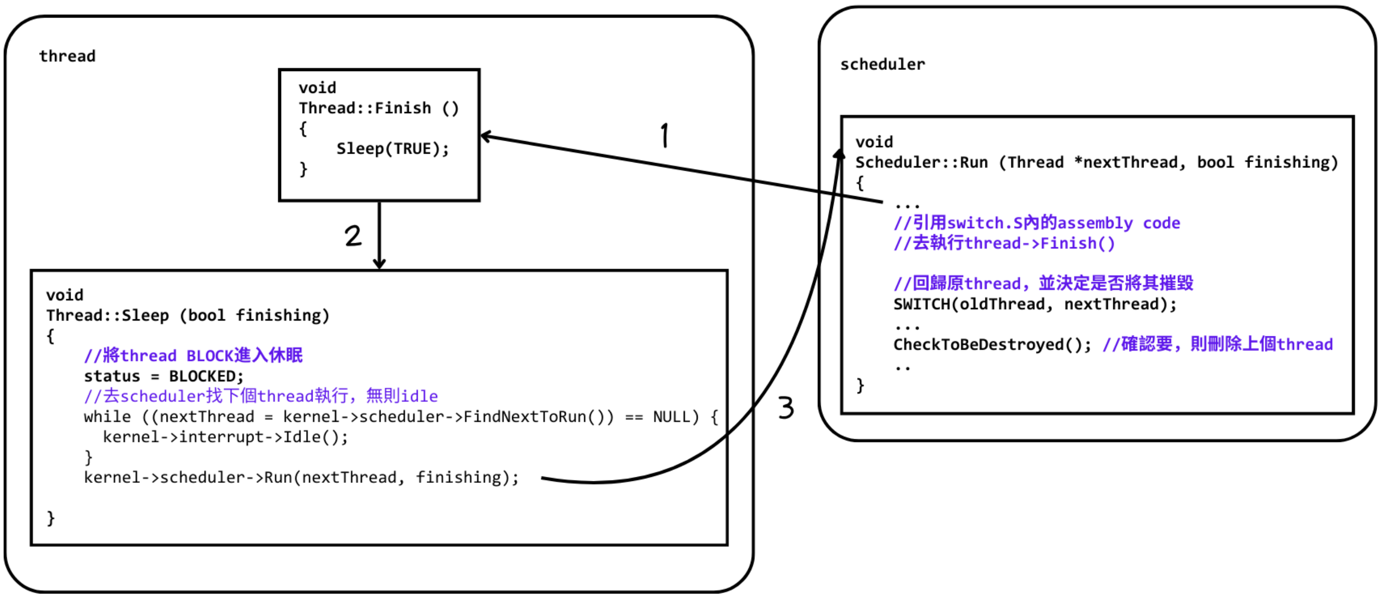
thread

addrspace

```
bool
AddrSpace::Load(char *fileName)
{
  //計算process所需資料的大小
  //分配page給process
  //將code, initdata, stack等資料讀入page
}
```

```
void
AddrSpace::Execute(char* fileName)
{
  //放好addrspace, reg等，而後正式執行
  kernel->machine->Run();
}
```

finish



```
void
Thread::Finish ()
{
    Sleep(TRUE);
}
```

```
void
Thread::Sleep (bool finishing)
{
    //將thread BLOCK進入休眠
    status = BLOCKED;
    //去scheduler找下個thread執行，無則idle
    while ((nextThread = kernel->scheduler->FindNextToRun()) == NULL) {
      kernel->interrupt->Idle();
    }
    kernel->scheduler->Run(nextThread, finishing);

}
```

```
void
Scheduler::Run (Thread *nextThread, bool finishing)
{
    ...
    //引用switch.S內的assembly code
    //去執行thread->Finish()

    //回歸原thread，並決定是否將其摧毀
    SWITCH(oldThread, nextThread);
    ...
    CheckToBeDestroyed(); //確認要，則刪除上個thread
    ..
}
```

Part 2:Implementation

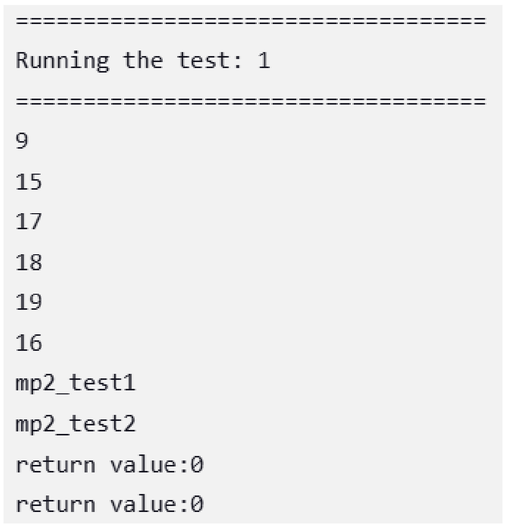
1. Detail of your implementation

實作 multiprogramming 前：

會分配所有 frame 給單一 process，如有多個 process，則會存取到其他 process 的 page，誤用 code section，造成 output 混亂、無法執行完成。

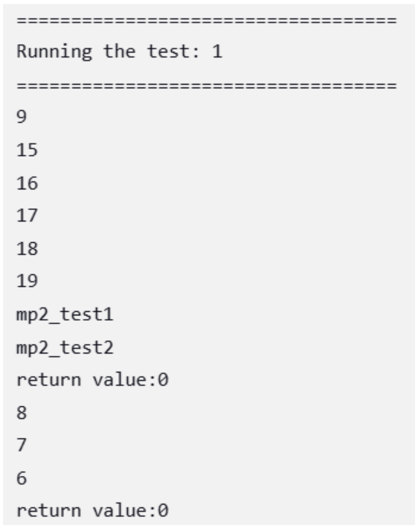
實作 multiprogramming 後：

系統會分配不重複的 frame 給各個 process，使其能不互相干擾的。



```
====================================
Running the test: 1
====================================
9
15
17
18
19
16
mp2_test1
mp2_test2
return value:0
return value:0
```

實作 multiprogramming 前

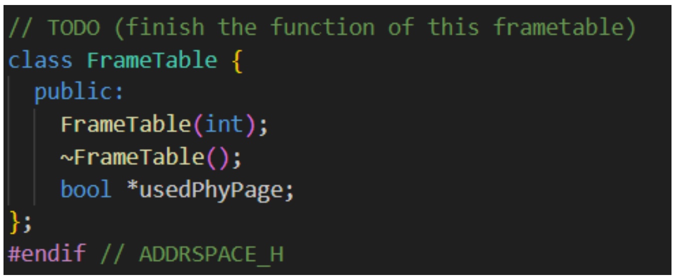


```
===================================
Running the test: 1
===================================
9
15
16
17
18
19
mp2_test1
mp2_test2
return value:0
8
7
6
return value:0
```

實作 multiprogramming 後

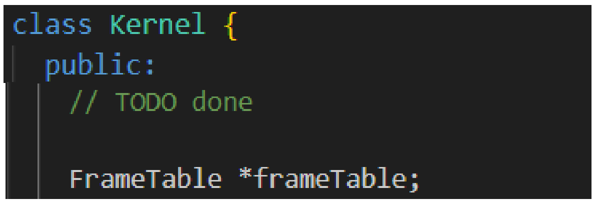
改動：
在 addrspace.h 底下新定義一個 data structrure "FrameTable"，並讓 kernel 在初始化時建立，紀錄已被分配的 frame。

addrspace.h



```
// TODO (finish the function of this frametable)
class FrameTable {
  public:
    FrameTable(int);
    ~FrameTable();
    bool *usedPhyPage;
};
#endif // ADDRSPACE_H
```

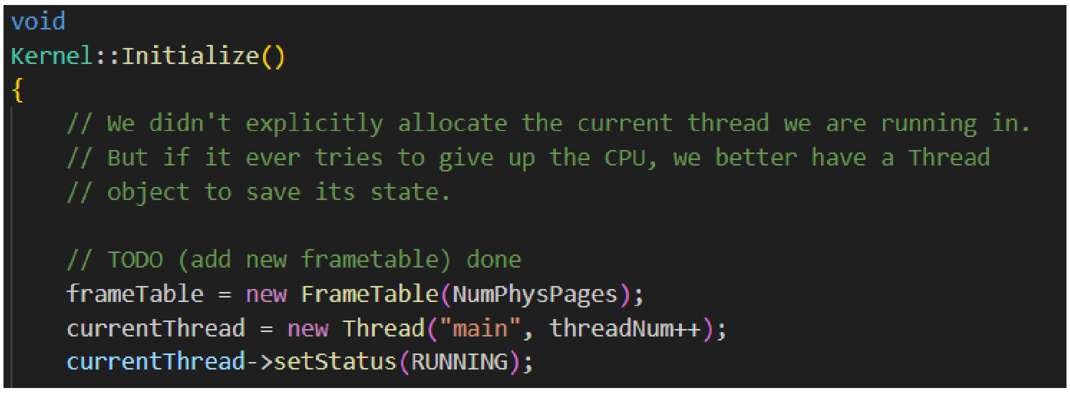
kernel.h



```
class Kernel {
  public:
    // TODO done

    FrameTable *frameTable;
```

kernel.cc

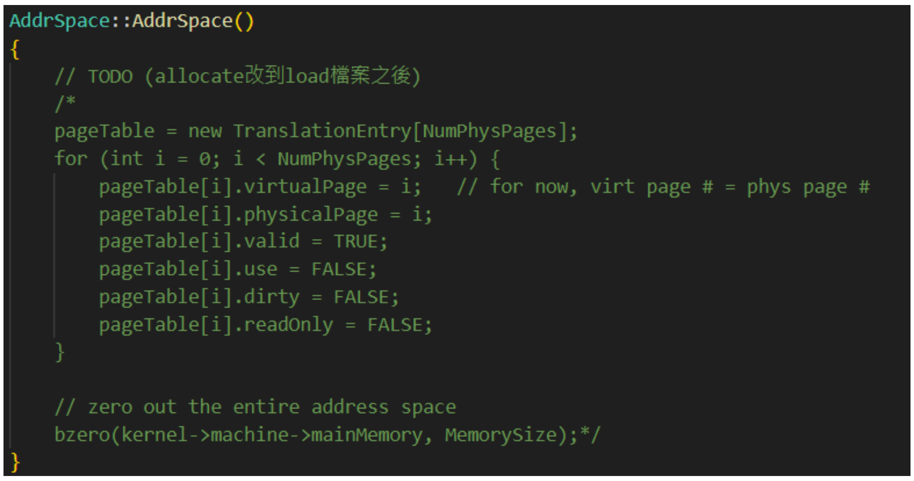


```
void
Kernel::Initialize()
{
    // We didn't explicitly allocate the current thread we are running in.
    // But if it ever tries to give up the CPU, we better have a Thread
    // object to save its state.

    // TODO (add new frametable) done
    frameTable = new FrameTable(NumPhysPages);
    currentThread = new Thread("main", threadNum++);
    currentThread->setStatus(RUNNING);
```

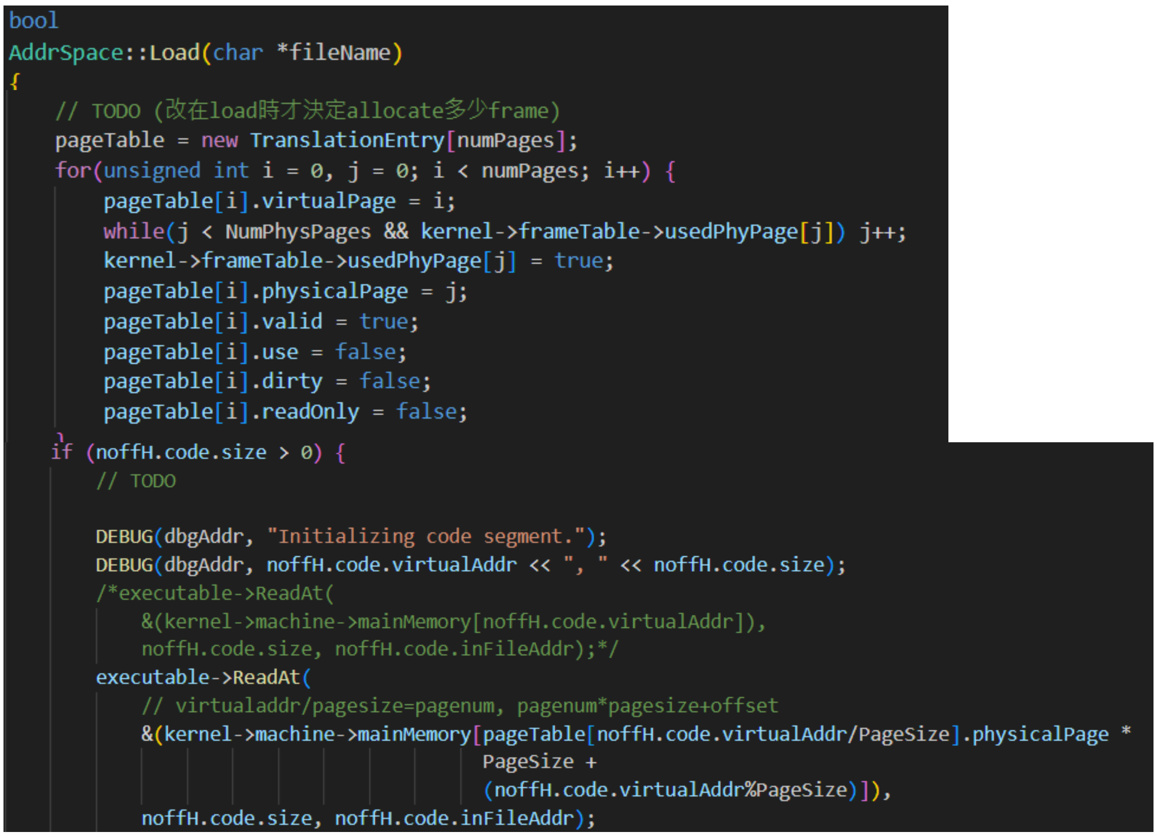
初次建立 addrspace 時，不將所有 frame 分配給該 process，我們將分配步驟推遲到 load，確認 process 所需的 frame 數後，再依其進行不重複的分配。而因為 virtual page number 不再等於 physical page number，故讀寫&釋放需仰賴 page table 進行 mapping。

addrspace.cc



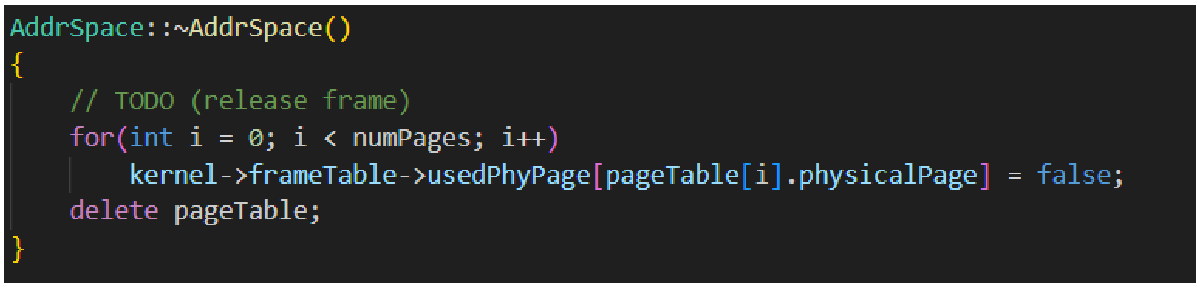
```
AddrSpace::AddrSpace()
{
    // TODO (allocate改到load檔案之後)
    /*
    pageTable = new TranslationEntry[NumPhysPages];
    for (int i = 0; i < NumPhysPages; i++) {
        pageTable[i].virtualPage = i;   // for now, virt page # = phys page #
        pageTable[i].physicalPage = i;
        pageTable[i].valid = TRUE;
        pageTable[i].use = FALSE;
        pageTable[i].dirty = FALSE;
        pageTable[i].readOnly = FALSE;
    }

    // zero out the entire address space
    bzero(kernel->machine->mainMemory, MemorySize);*/
}
```



```
bool
AddrSpace::Load(char *fileName)
{
    // TODO (改在load時才決定allocate多少frame)
    pageTable = new TranslationEntry[numPages];
    for(unsigned int i = 0, j = 0; i < numPages; i++) {
        pageTable[i].virtualPage = i;
        while(j < NumPhysPages && kernel->frameTable->usedPhyPage[j]) j++;
        kernel->frameTable->usedPhyPage[j] = true;
        pageTable[i].physicalPage = j;
        pageTable[i].valid = true;
        pageTable[i].use = false;
        pageTable[i].dirty = false;
        pageTable[i].readOnly = false;
    }
    if (noffH.code.size > 0) {
        // TODO

        DEBUG(dbgAddr, "Initializing code segment.");
        DEBUG(dbgAddr, noffH.code.virtualAddr << ", " << noffH.code.size);
        /*executable->ReadAt(
            &(kernel->machine->mainMemory[noffH.code.virtualAddr]),
            noffH.code.size, noffH.code.inFileAddr);*/
        executable->ReadAt(
            // virtualaddr/pagesize=pagenum, pagenum*pagesize+offset
            &(kernel->machine->mainMemory[pageTable[noffH.code.virtualAddr/PageSize].physicalPage *
                                          PageSize +
                                          (noffH.code.virtualAddr%PageSize)]),
            noffH.code.size, noffH.code.inFileAddr);
```



```
AddrSpace::~AddrSpace()
{
    // TODO (release frame)
    for(int i = 0; i < numPages; i++)
        kernel->frameTable->usedPhyPage[pageTable[i].physicalPage] = false;
    delete pageTable;
}
```

Part 3:Contribution

1. Describe details and percentage of each member's contribution.

姓名	負責項目	貢獻度
何翊華	Trace code、Implementation、Report	50%
廖偉辰	Trace code、Implementation、Report	50%