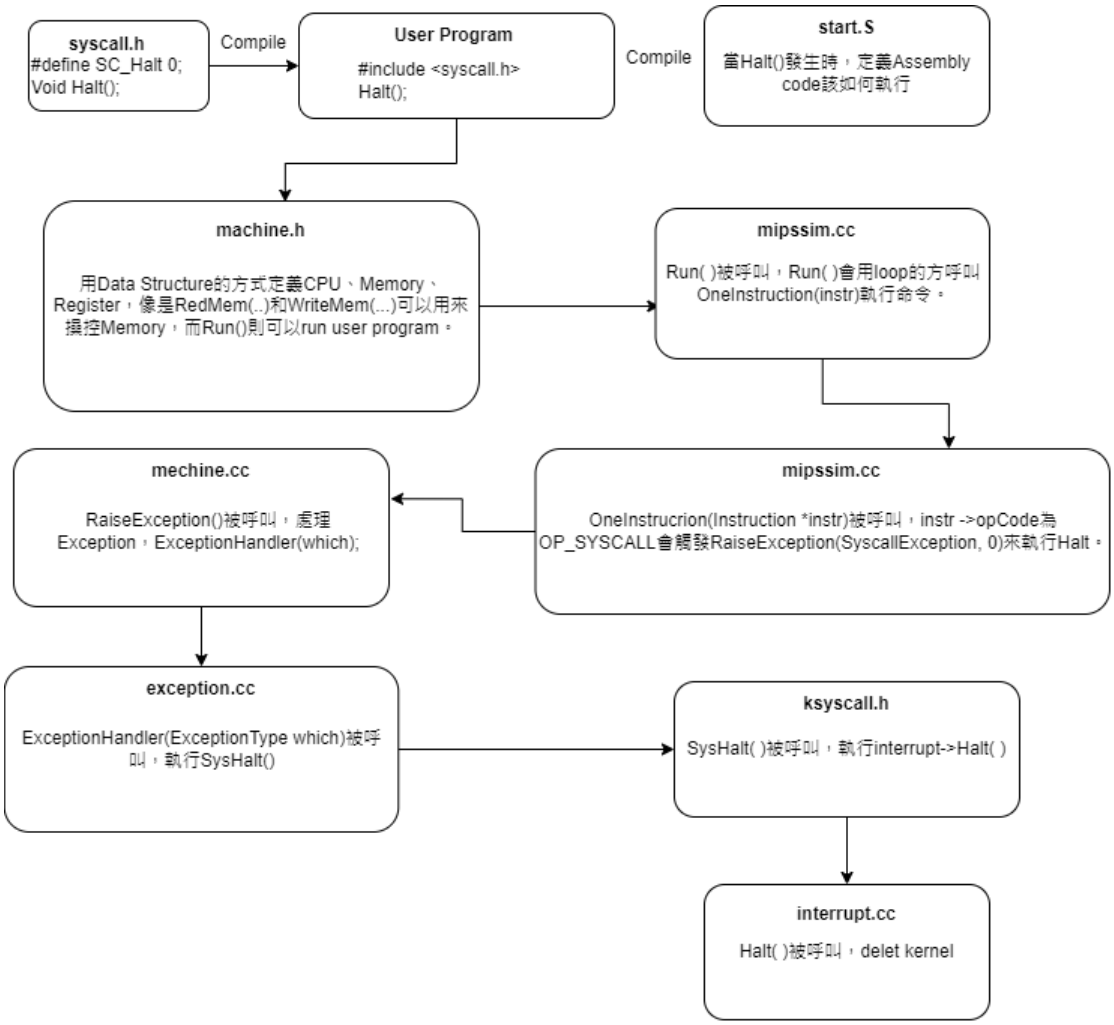
FSS HW00 111598034
郭駿頤

Part 1：Trace Code Result

1. Flow Chart of Halt() System Call:



2. Details of Trace Code

1. mipssim.cc

執行 Run( )

```
//----------------------------------------------------------------------

void
Machine::Run()
{
    Instruction *instr = new Instruction;  // storage for decoded instruction

    if (debug->IsEnabled('m')) {
        cout << "Starting program in thread: " << kernel->currentThread->getName();
        cout << ", at time: " << kernel->stats->totalTicks << "\n";
    }
    kernel->interrupt->setStatus(UserMode);
    for (;;) {
        OneInstruction(instr);
        kernel->interrupt->OneTick();
        if (singleStep && (runUntilTime <= kernel->stats->totalTicks))
            Debugger();
    }
}
```

當 user-level program 執行時，會呼叫 Run( )，Run( )會在 loop 裡呼叫 OneInstruction(instr)執行命令。

2. mipssim.cc
執行 OneInstruction(instr)

```
//----------------------------------------------------------------------

void
Machine::OneInstruction(Instruction *instr)
{
#ifdef SIM_FIX
    int byte;       // described in Kane for LWL,LWR,...
#endif
```

```
    break;

      case OP_SYSCALL:
    RaiseException(SyscallException, 0);
    return;

      case OP_XOR:
```

處理 Halt( )時，instr->opcode = OP_SYSCALL，所以會呼叫 RaiseException(SyscallException, 0)。

3. mechine.cc
執行 RaiseException(ExceptionType which, int badVAddr)。

```
//----------------------------------------------------------------------

void
Machine::RaiseException(ExceptionType which, int badVAddr)
{
    DEBUG(dbgMach, "Exception: " << exceptionNames[which]);
    registers[BadVAddrReg] = badVAddr;
    DelayedLoad(0, 0);          // finish anything in progress
    kernel->interrupt->setStatus(SystemMode);
    ExceptionHandler(which);        // interrupts are enabled at this point
    kernel->interrupt->setStatus(UserMode);
}
```

會呼叫 ExceptionHandler(which)進行處理。

4. exception.cc
執行 ExceptionHandler(ExceptionType which)。

```
void
ExceptionHandler(ExceptionType which)
{
    int type = kernel->machine->ReadRegister(2);
    int val;
    int status, exit, threadID, programID;
    DEBUG(dbgSys, "Received Exception " << which << " type: " << type << "\n");
    switch (which) {
    case SyscallException:
        switch(type) {
        case SC_Halt:
            DEBUG(dbgSys, "Shutdown, initiated by user program.\n");
            SysHalt();
                        cout<<"in exception\n";
            ASSERTNOTREACHED();
            break;
```

which 決定此次的 exception 是 system call ，type 存放 Register 的指令，以 Halt 為例，Register 存的是 SC_Halt，執行 SysHalt( )。

5. ksyscall.h

執行 SysHalt( )。

```
void SysHalt()
{
  kernel->interrupt->Halt();
}
```

會觸發 kernel->interrupt->Halt( )，真正執行 Halt 背後的程序。

6. interrupt.cc

執行 Halt( )。

```
//----------------------------------------------
void
Interrupt::Halt()
{
    // MP4 mod tag
    /*
    cout << "Machine halting!\n\n";
    cout << "This is halt\n";
    kernel->stats->Print();
    */
    delete debug;

    delete kernel;  // Never returns.
}
```

將 kernel 刪除。