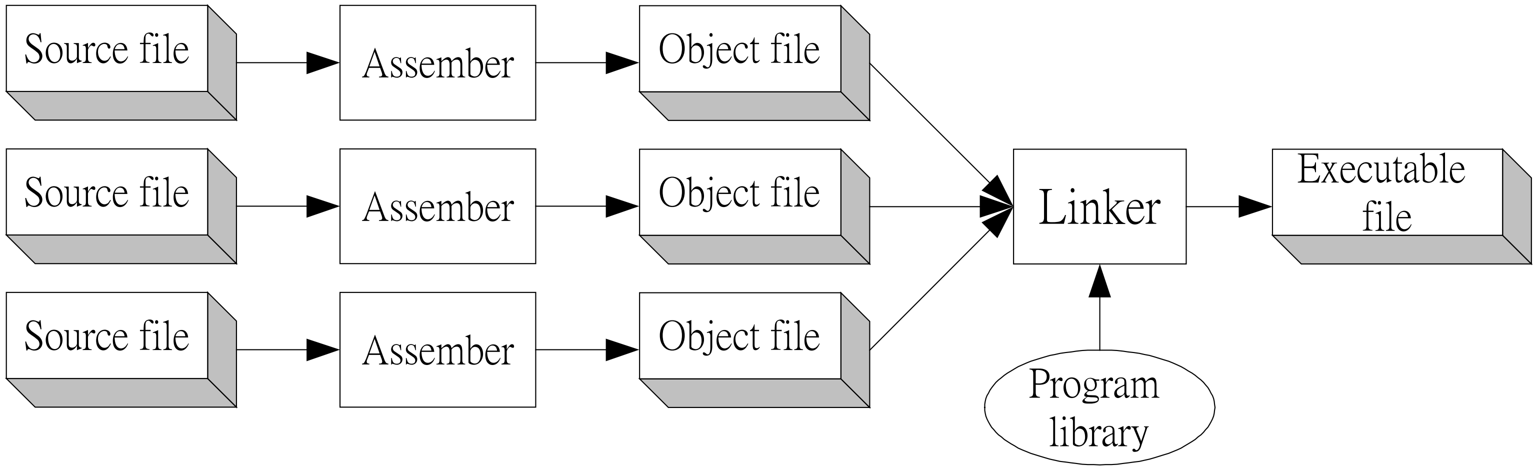
SPIM 簡介

陳嘉彬 陳玟丞

2002/9/25

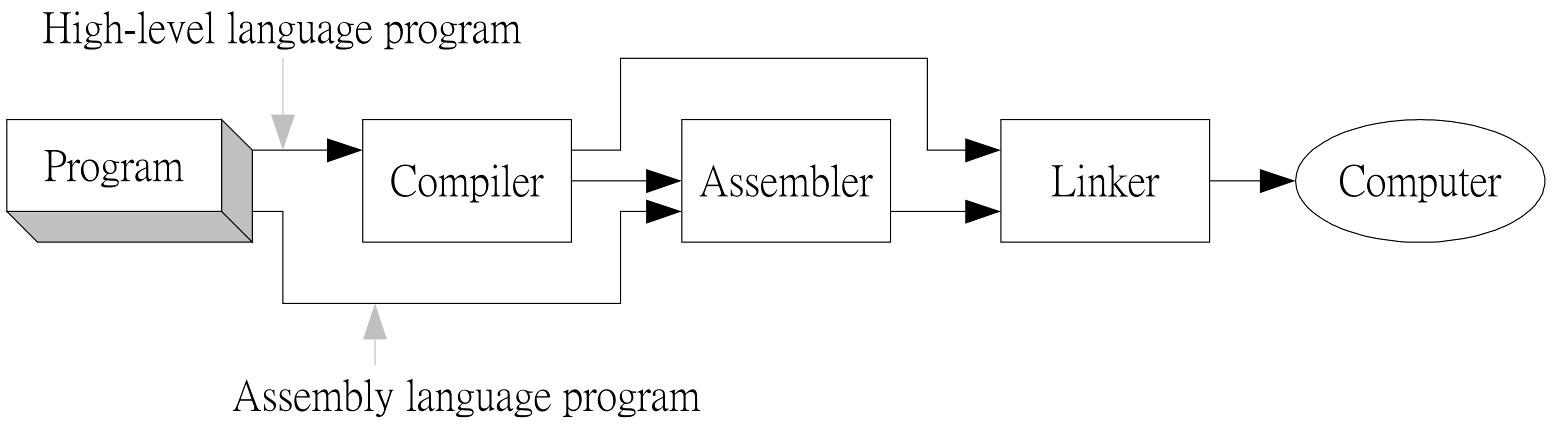
Introduction

- The process that produces an executable file



Introduction

- The primary reason to program in assembly language is that the speed or size of a program is critically important.



Assemblers-Local and Global Labels

- Labels are local by default and must be explicitly declared global.

```
                .text
                .globl  main
global  main:
                la   $s0, data0
        ..................................
local   loop:
        ..................................

                .data
local   data0:  .word           1,3
        str0:   .asciiz         “The sum is ”
```

Assemblers-Marcos

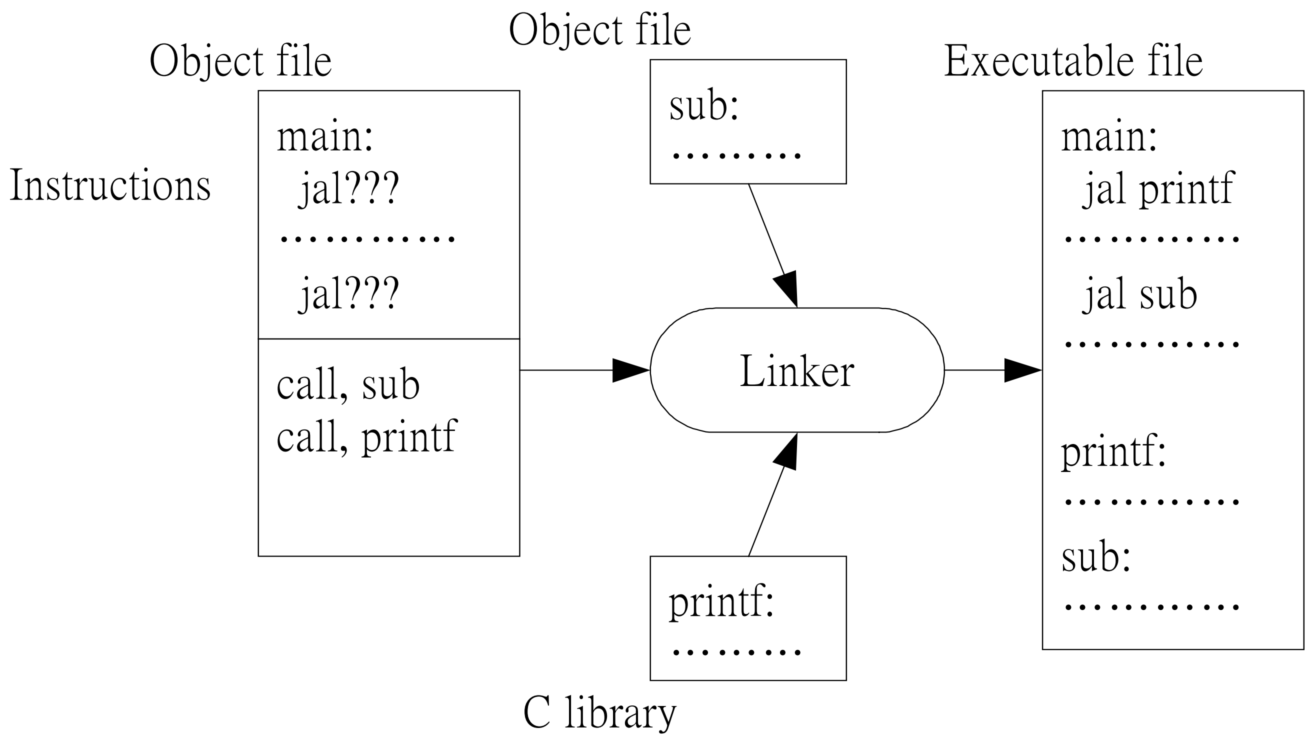
```
            .data
int_str:    .asciiz             "%d "

            .text
            la     $a0, int_str
            mov    $a1, $7
            jal    printf
..............................
..............................
            la     $a0, int_str
            mov    $a1, $t0
            jal    printf
```

Assemblers-Marcos

```
         .data
int_str: .asciiz          "%d "

         .text
         .macro   print_int($arg)
         la       $a0, int_str
         mov      $a1, $arg
         jal      printf
         .end_macro


         print_int ($7)
         print_int ($t0)
         print_int ($a0)
```

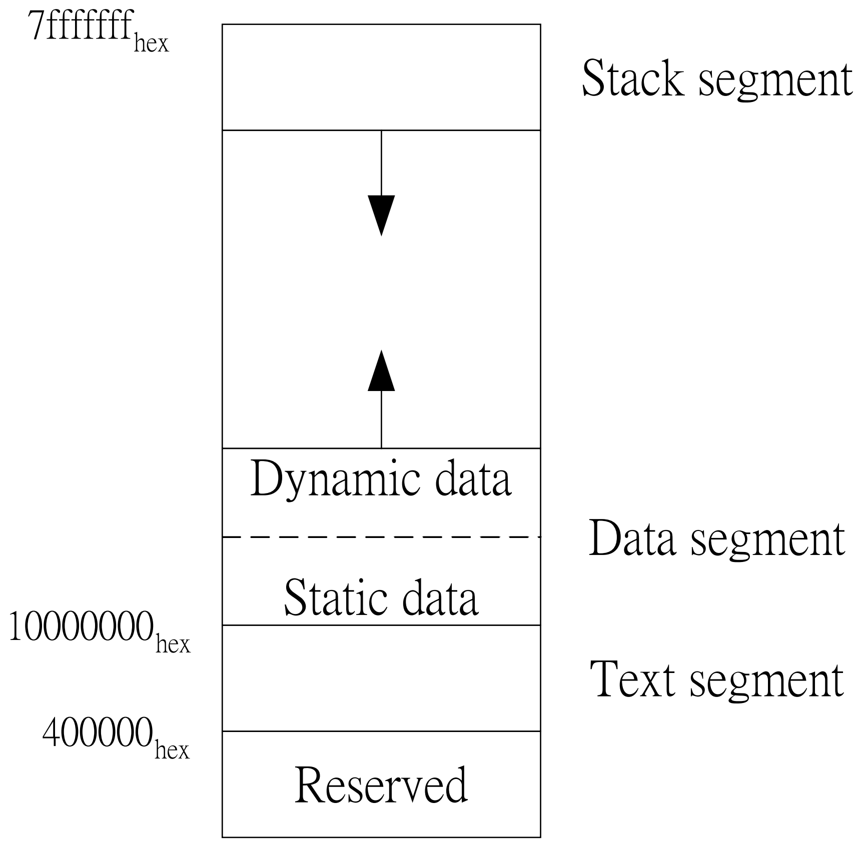
```
la    $a0, int_str
mov   $a1, $a0
jal   printf
```

Linker


Object file
Instructions
main:
jal???
…………
jal???
call, sub
call, printf
Object file
sub:
………
Linker
printf:
………
C library
Executable file
main:
jal printf
…………
jal sub
…………
printf:
…………
sub:
…………

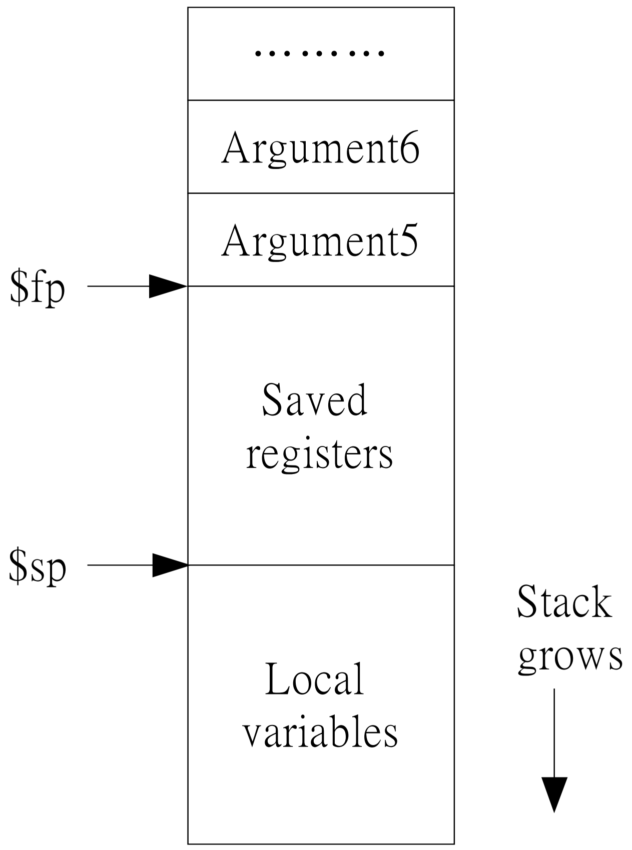
Memory Usage

- The maximum size of a program's stack and dynamic data are unknown. The operating system expands them to meet demand.



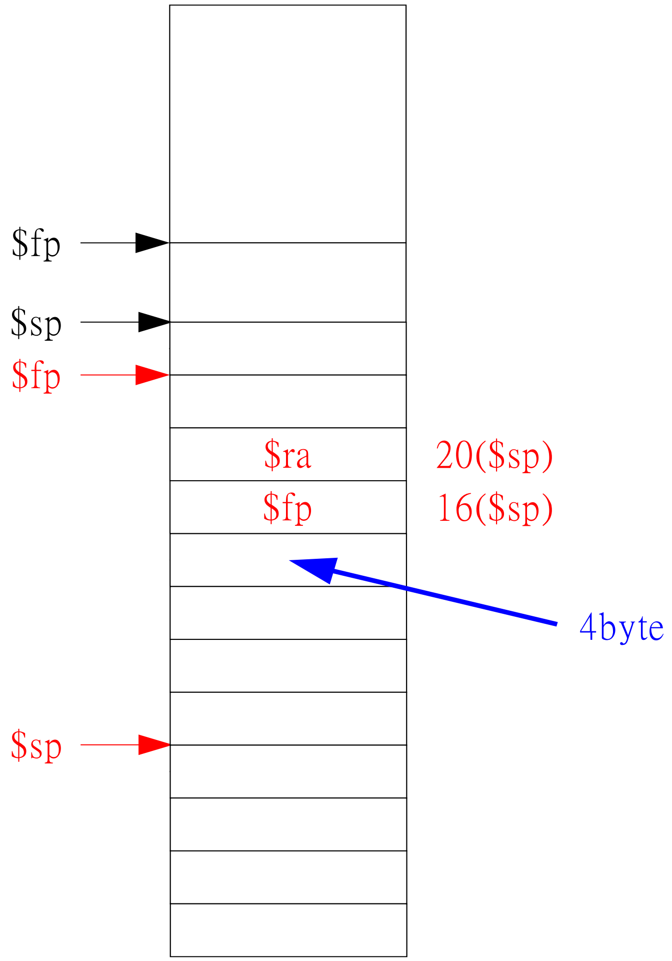
Procedure Calls

- Procedure calls and returns follow a strict last-in, first-out (LIFO) order, so this memory can be allocated and deallocated on a stack
- Frame pointer ($fp) :
  - first word of the frame
- Stack pointer ($sp) :
  - last word of the frame



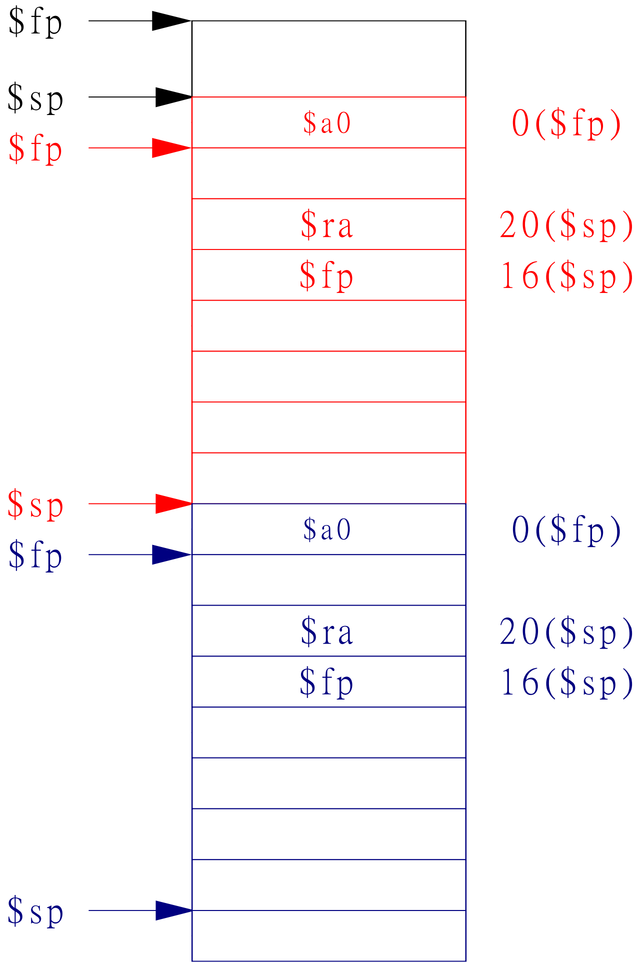
Procedure Calls

```
          .text
          .globl  main
main:

          subu      $sp, $sp, 32
          sw        $ra, 20($sp)
          sw        $fa, 16($sp)
          addu      $fp, $sp, 28

          li        $a0, 10
          jal       fact

          la        $a0,$LC
....................................
....................................
```



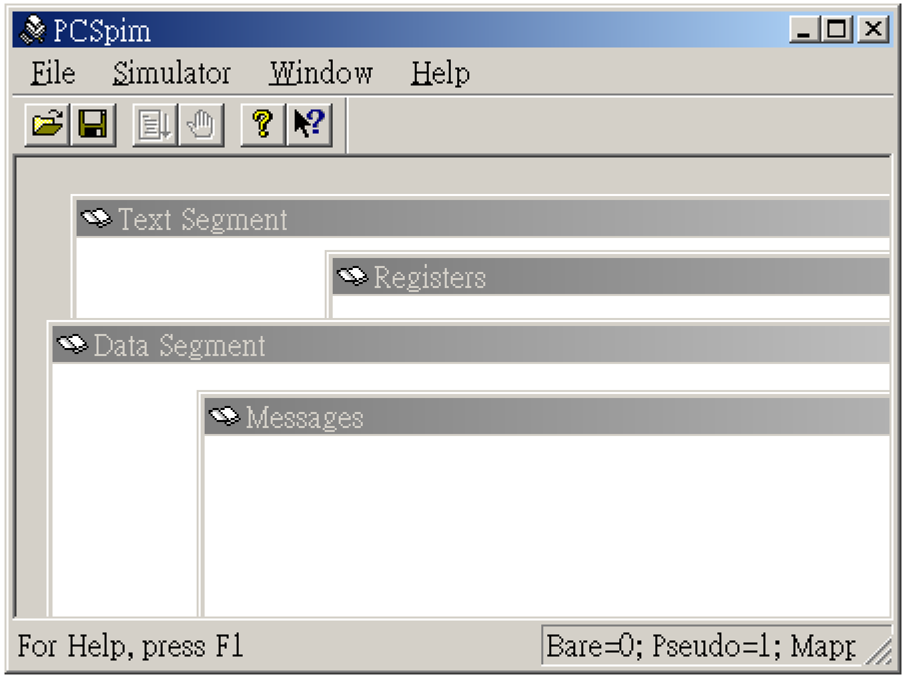
Procedure Calls

```
fact:

        subu    $sp, $sp, 32
        sw      $ra, 20($sp)
        sw      $fa, 16($sp)
        addu    $fp, $sp, 28

        sw      $a0, 0($fp)
.................................
.................................
        lw      $ra, 20($sp)
        lw      $fa, 16($sp)
        addu    $sp, $sp, 32
        j       $ra
```



Setup SPIM Simulator

- Download spim.zip
  - spimwin.exe
  - sort.asm
  - MIPS攻略.doc
  - 計組2002.ppt
- Execute spimwin.exe
- Execute pcspim.exe

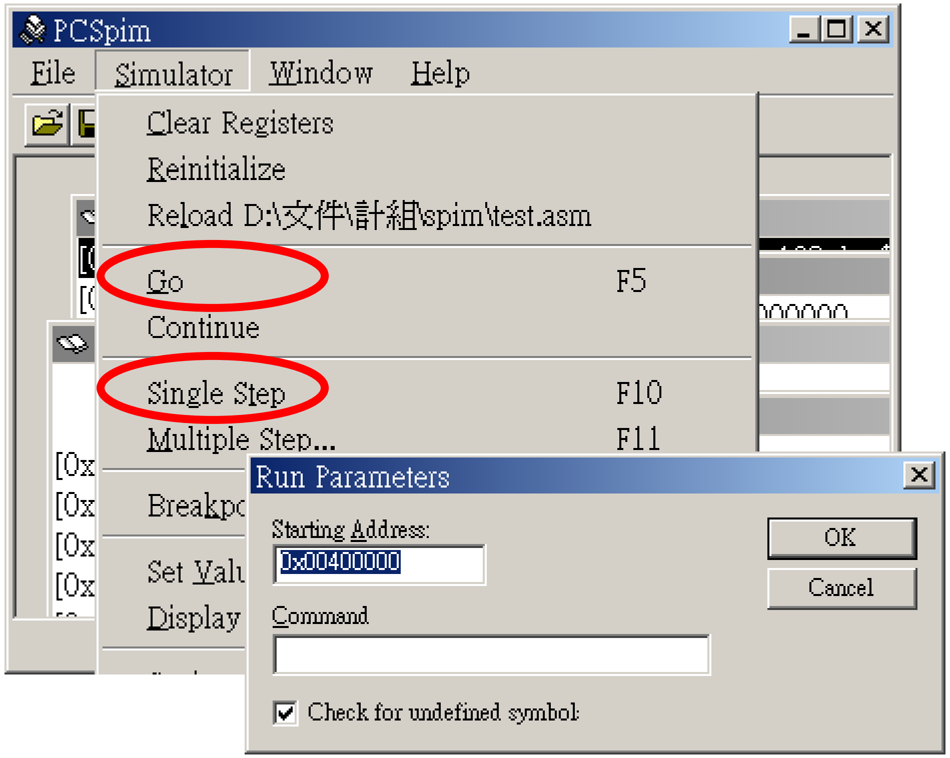


For Help, press F1

Bare=0; Pseudo=1; Mapp

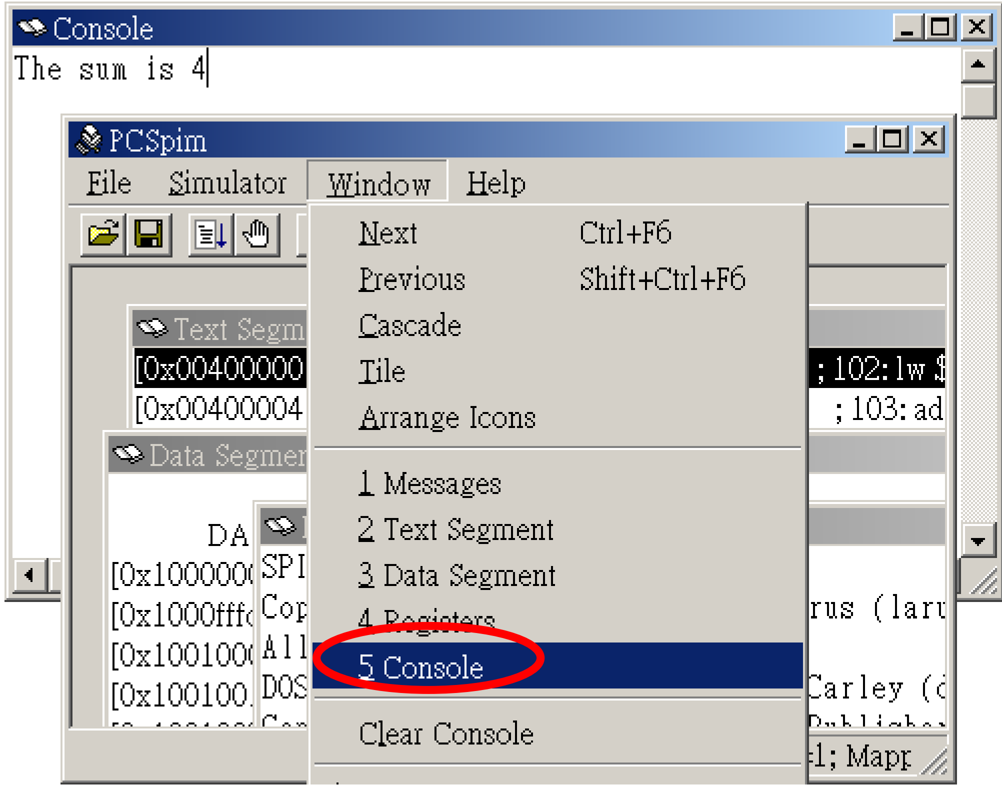
SPIM User's Guide

- Open File (filename.asm)
- Simulator
  - Single Step
  - Go
    - set starting address
    - ok



SPIM User's Guide

- Display Results
  - Window
    - Console



```
The sum is 4
```

- Save File (Pcspim.log)

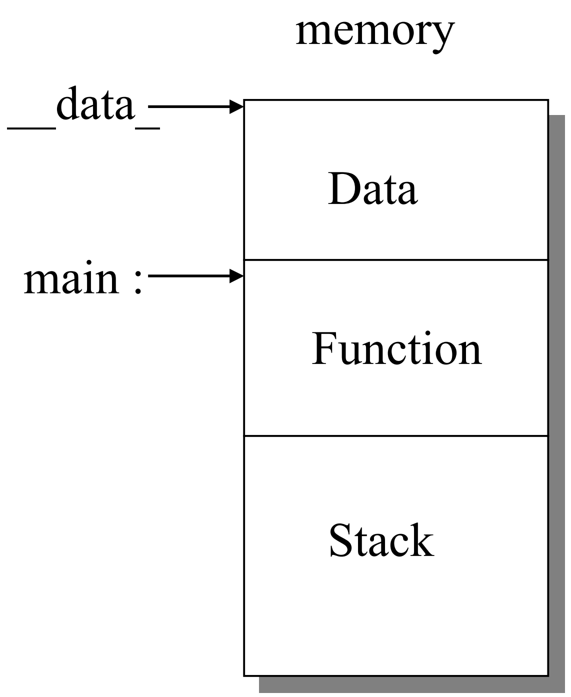
```
Console
===============
The sum is 4
```

How to Program

- A assembly program could divide three parts
  - Initial setup : arrange data and program into data memory and program memory
  - Main function : the most important section of your program
  - Sub function : subroutine 、

Initial Setup

```
            .data
__data_:    .word     1,3
__txt1_:    .asciiz   "The sum is "

            .text
            .globl  main
main:


            ..............................
            ..............................
```



Main Function

```
main:
     li      $v0 4
     la      $a0 __txt1  #.......
     syscall
     la      $a0 __data
     li      $a1 5          # comment
     jal     display

       ............................
       ............................
       ............................
```

1. Use label well
2. Remember to write comments
3. Main function must be

   brief and clear.

Sub Function

```
main:
      li   $v0 4
      la   $a0 __txt1    #.......
      j    sort

sort :
        addi    $sp , $sp , -20
        sw      $s3 , 16($sp)
        move  $s2 , $ a1
        lw      ………..
        ………………….
```

1. Use label well
2. Remember to write comments

System Calls

Service	System call code	Arguments	Result
print_int	1	$a0=integer	
print_float	2	$f12=float	
print_double	3	$f12=double	
print_string	4	$a0=string	
read_int	5		integer(in $v0)

System Calls

```
            .data
data0:      .word               1,3
str0:       .asciiz             "The sum is "

            .text
            .globl  main
main:
            li   $v0, 4                   # 列印字串
            la   $a0, str0                # 字串名稱爲str
            syscall

            la   $s0 data0
            lw   $a0, 0($s0)

            li   $v0 1                    # 列印($a0)所儲存的整數
            syscall
            li   $v0 10                   # 結束syscall
            syscall
```

System Calls

```
        .data
data0:  .word        3
        .text
        .globl  main
main:

        la   $s0 data0
        lw   $a0, 0($s0)

        li   $v0 5                  # 輸入整數到($v0)
        syscall
        add  $a0,$a0,$v0

        li   $v0 1                  # 列印($a0)所儲存的整數
        syscall
        li   $v0 10                 # 結束syscall
        syscall
```