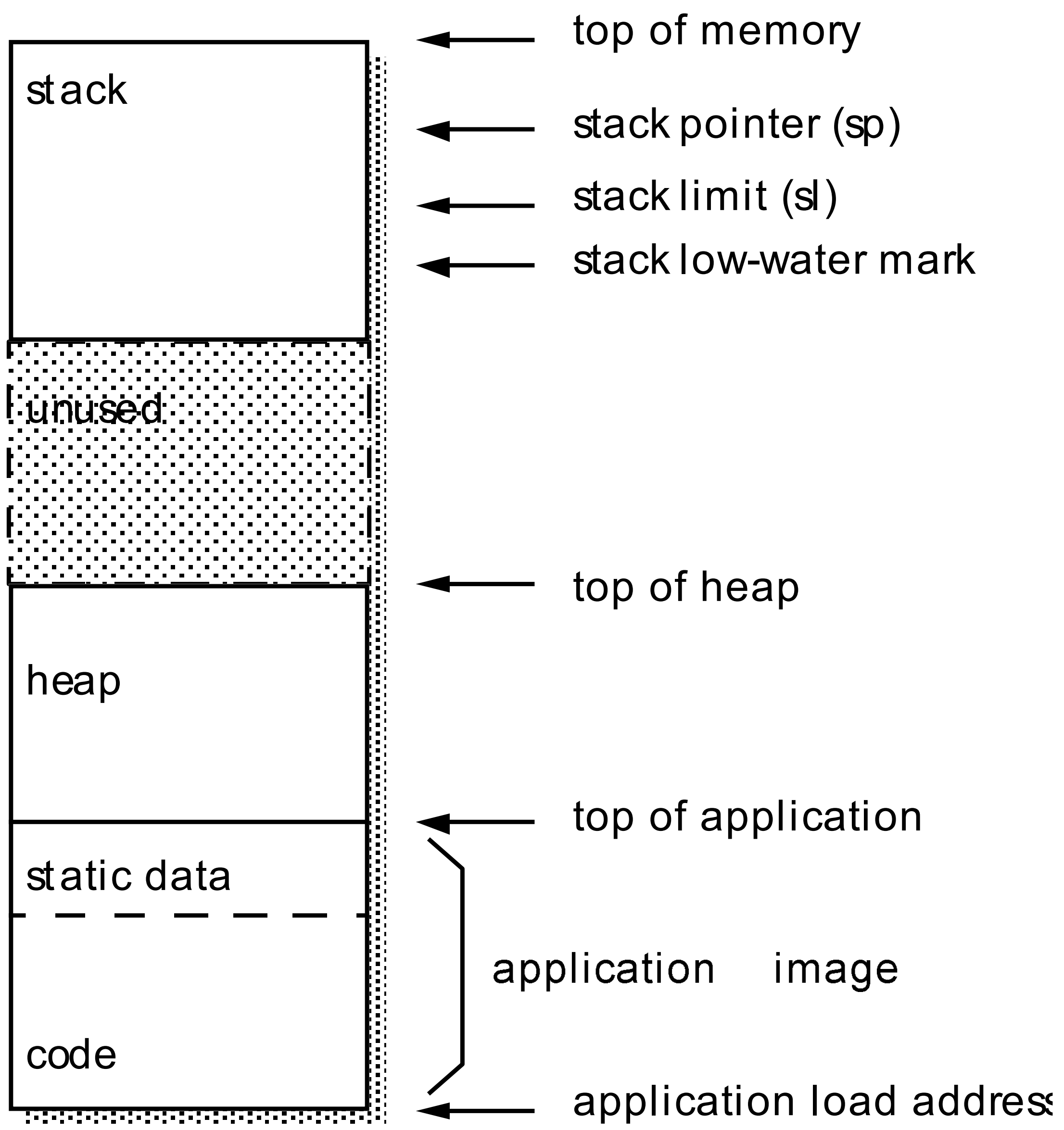
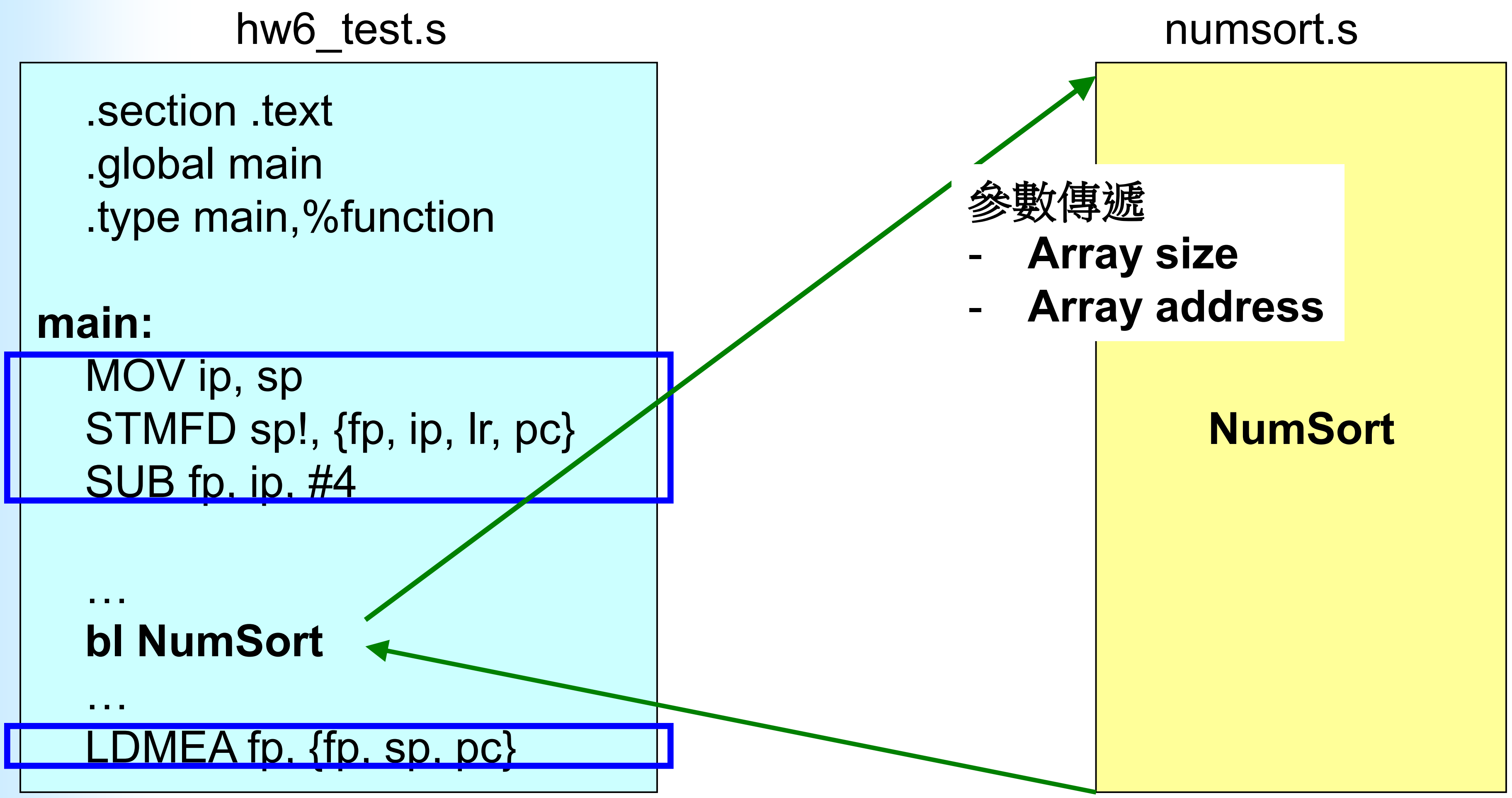
Homework #6 (1)

- Write a function called **NumSort** to sort an integer array from the smallest to the biggest.
- Two arguments will be passed into your function by stack
  - **Array size**
  - **The address of the first element in array**
- **The result of NumSort**
  - The result array in which each element is sorted from the smallest to the biggest. (原來的integer array裡的沒有被修改，只是讀取原integer array，並排序好的結果存放於result array)
  - Register r8 will have the address of the result array.

Homework #6 (2)

- Ex: an integer array=[1,10,6,3,20,40,9]
  - Result: **1, 3, 6, 9, 10, 20,40**
- Ex: an integer array=[12,4,2,45,23,8,50,67]
  - Result: **2, 4, 8,12, 23, 45, 50, 67**


top of memory
stack pointer (sp)
stack limit (sl)
stack low-water mark
stack
unused
top of heap
heap
top of application
static data
application image
code
application load addres

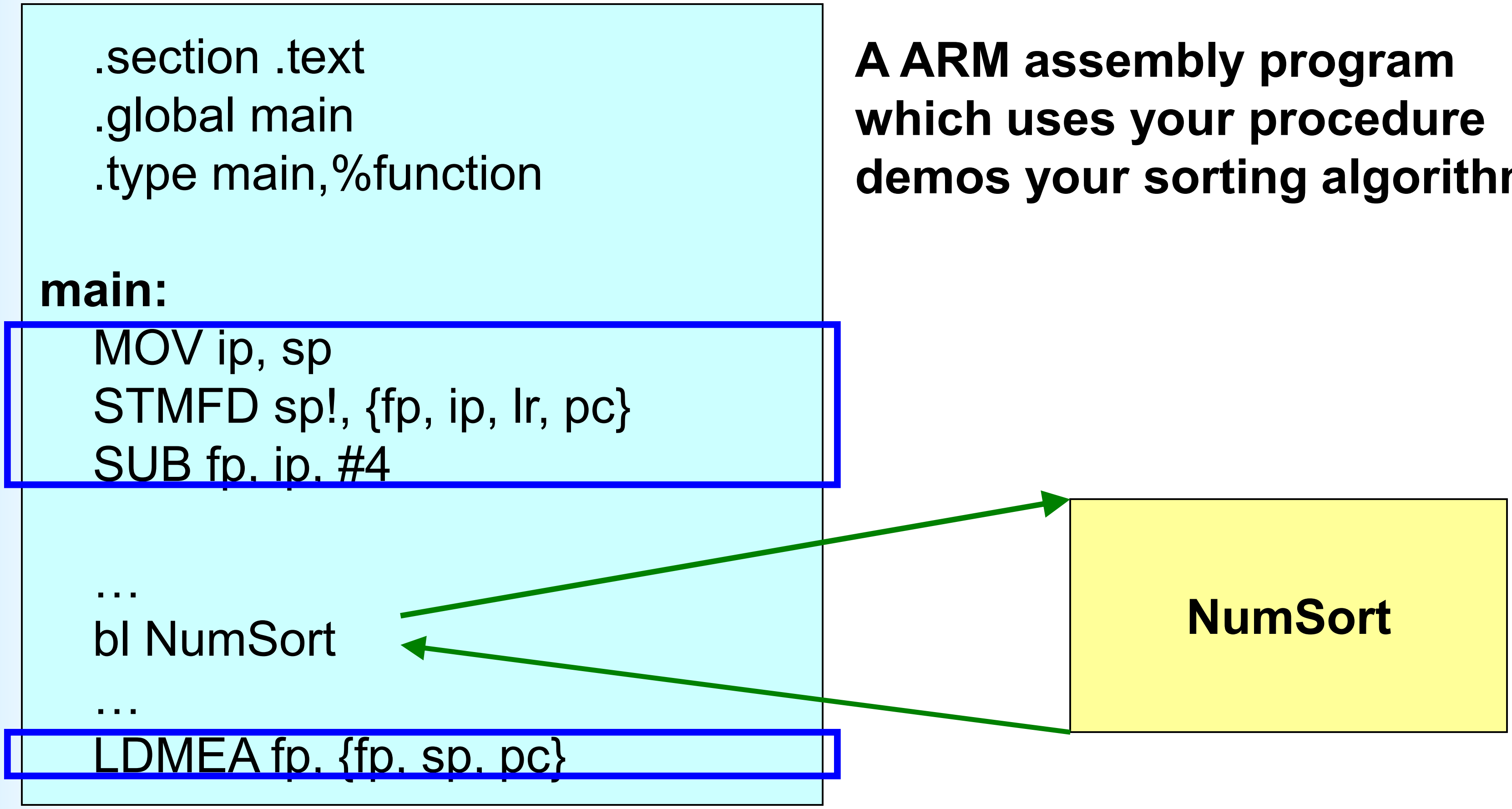


```
    .section .text
    .global main
    .type main,%function

main:
    MOV ip, sp
    STMFD sp!, {fp, ip, lr, pc}
    SUB fp, ip, #4

    …
    bl NumSort
    …
    LDMEA fp, {fp, sp, pc}
```

Homework #6 (3)



```
    .section .text
    .global main
    .type main,%function

main:
    MOV ip, sp
    STMFD sp!, {fp, ip, lr, pc}
    SUB fp, ip, #4

    …
    bl NumSort
    …
    LDMEA fp, {fp, sp, pc}
```

A ARM assembly program which uses your procedure demos your sorting algorithm

hw6_test.s

```
 .section .text
        .global main
        .type main,%function
main:
        MOV ip, sp
        STMFD sp!, {fp, ip, lr, pc}
        SUB fp, ip, #4

        /* --- begin of your function --- */
        MOV r0, #100    /* array address */
        MOV r1, #200    /* array size */

        STR r0, [sp, #-4]!    /* push array address */
        STR r1, [sp, #-4]!    /* push array size */

        bl  NumSort
        /* --- end of your function --- */

        LDMEA fp, {fp, sp, pc}
        .end
```

Push array address and array size into stack

Homework #6 (4)

numsort.s

```
.section .text
        .global NumSort
        .type NumSort,%function

NumSort:
        /* function start */
        MOV ip, sp
        STMFD sp!, {r0-r10, fp, ip, lr, pc}
        SUB fp, ip, #4

        /* --- begin your function --- */
        LDR r5, [ip], #4  /* get array size */
        LDR r6, [ip], #4  /* get array address */

        /* DO NumSort */

        /* --- end of your function --- */
        nop
        /* function exit */
        LDMEA fp, {r0-r10, fp, sp, pc}
        .end
```

參數傳遞

Write your function

當執行到這裡時，r8應該要指向result array的位址

How to Compile Your Program?

- %arm-elf-gcc –g hw6_test.s numsort.s –o hw6.exe

Homework #6 (5)

- Program should be assembled and linked by gcc (ARM-ELF format)
  - 使用於作業一所安裝完成的cross compiler與cross binutils
- Program should be executed under **GDB ARM simulator**
- 程式中應有適當的說明（註解）
- You should turn in to **ECOURSE**
  - “**README.txt**” file: 文字檔，描述你程式的內容、如何編譯程式、如何執行你的程式
  - Your ARM assembly procedure，**檔名為：numsort.s**
  - An ARM assembly program which uses your NumSort procedure，**檔名為：hw6_test.s**
  - Any file needed in your work (ex: Makefile)
- **Deadline: December 6 (Sunday), 2015**