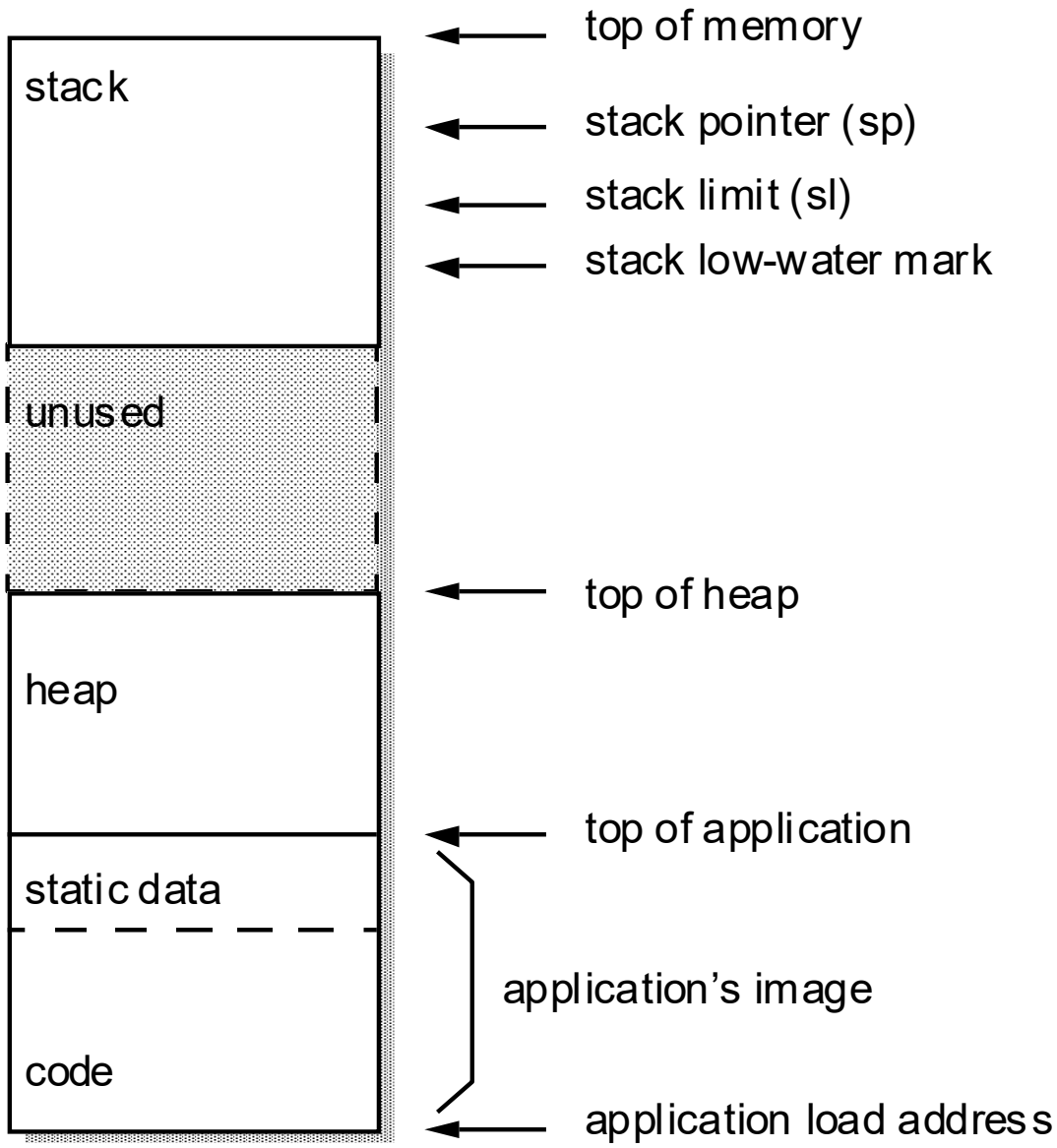
Homework #4 (1)

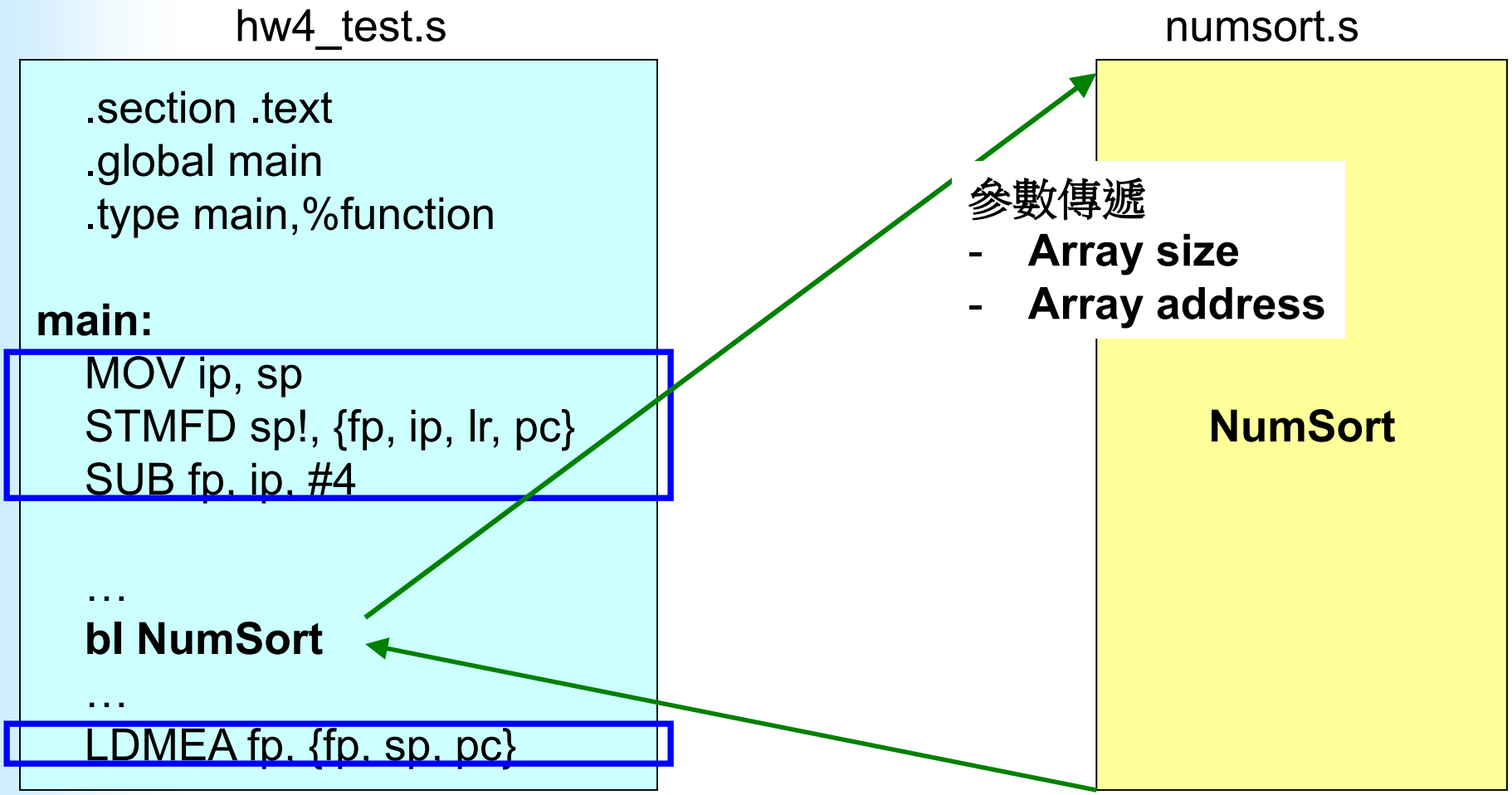
- Write a function called **NumSort** to sort an integer array from the biggest to the smallest.
- Two arguments will be passed into your function by stack
  - **Array size**
  - **The address of the first element in array**
- **The result of NumSort**
  - The result array in which each element is sorted from the biggest to the smallest. (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 r3 will have the address of the result array.

Homework #4 (2)

- Ex: an integer array=[1,10,6,3,20,40,9]
  - Result: **40, 20, 10, 9, 6, 3, 1**

- Ex: an integer array=[12,4,2,45,23,8,50,67]
  - Result: **67, 50, 45, 23, 12, 8, 4, 2**


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


hw4_test.s
.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}
numsort.s
參數傳遞
- Array size
- Array address
NumSort

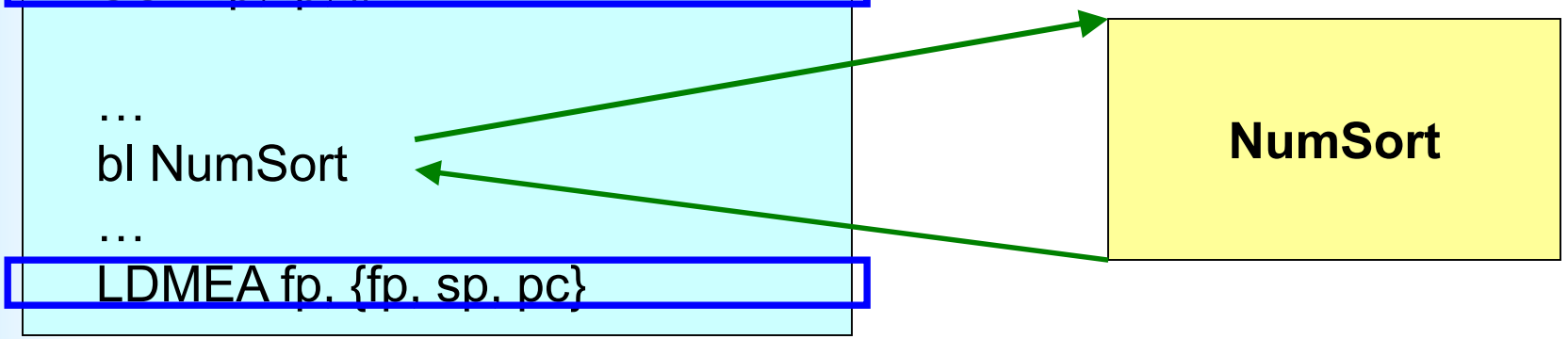
Homework #4 (3)

```
    .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}
```

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



hw4_test.s

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

    /* put array size into r0 */
    /* put array address into r1 */

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

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

- and array size => r0
- array address => r1

Homework #4 (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 --- */
    /* put array size into r0 */
    /* put array address into r1 */

    /* 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，r3應該要指向result array的位址

How to Compile Your Program?

- $arm-none-eabi-gcc –g hw4_test.s numsort.s –o hw4.exe

Homework #4 (5)

- Program should be assembled and linked by gcc
  - 使用於作業一所安裝完成的cross toolchain.
- 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 **ECOURSE2**
  - "**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：hw4_test.s**
  - Makefile / any file needed in your work
  - 請將欲繳交的檔案壓縮成 <**hw4_學號.tar.bz2**>，上傳壓縮檔
- **Deadline: November 20 (Friday), 2020**