

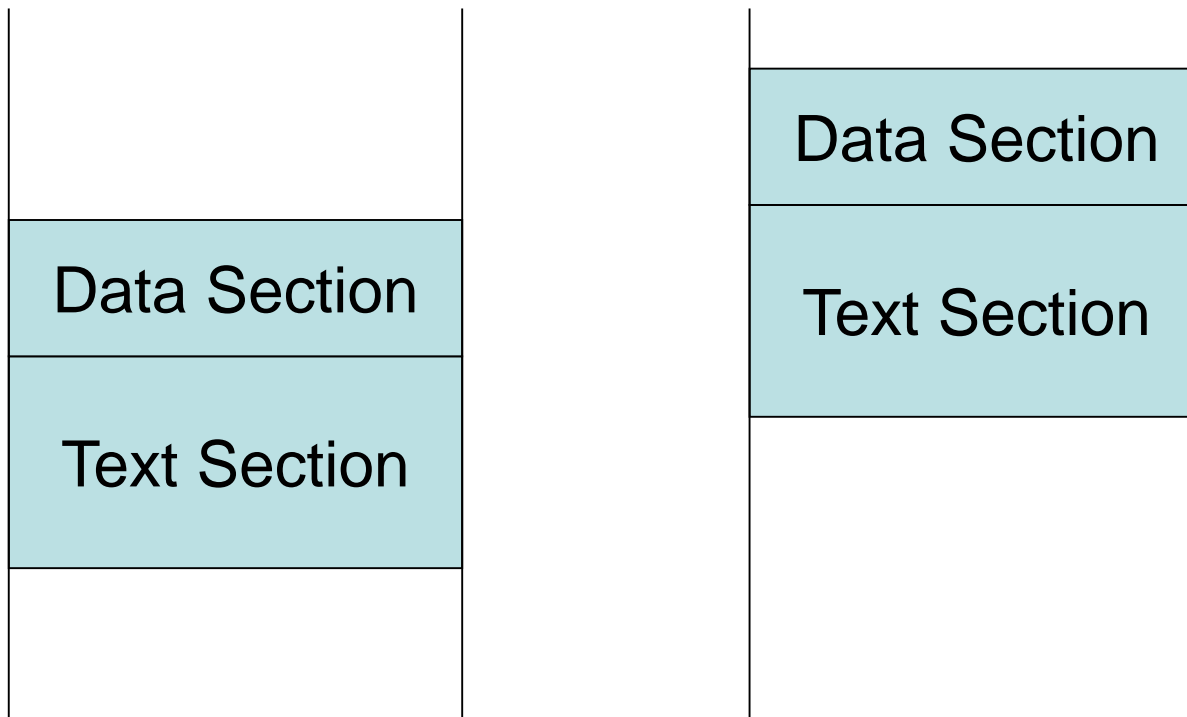
# **Position Independent Code Position Independent Data**

**Peng-Sheng Chen**

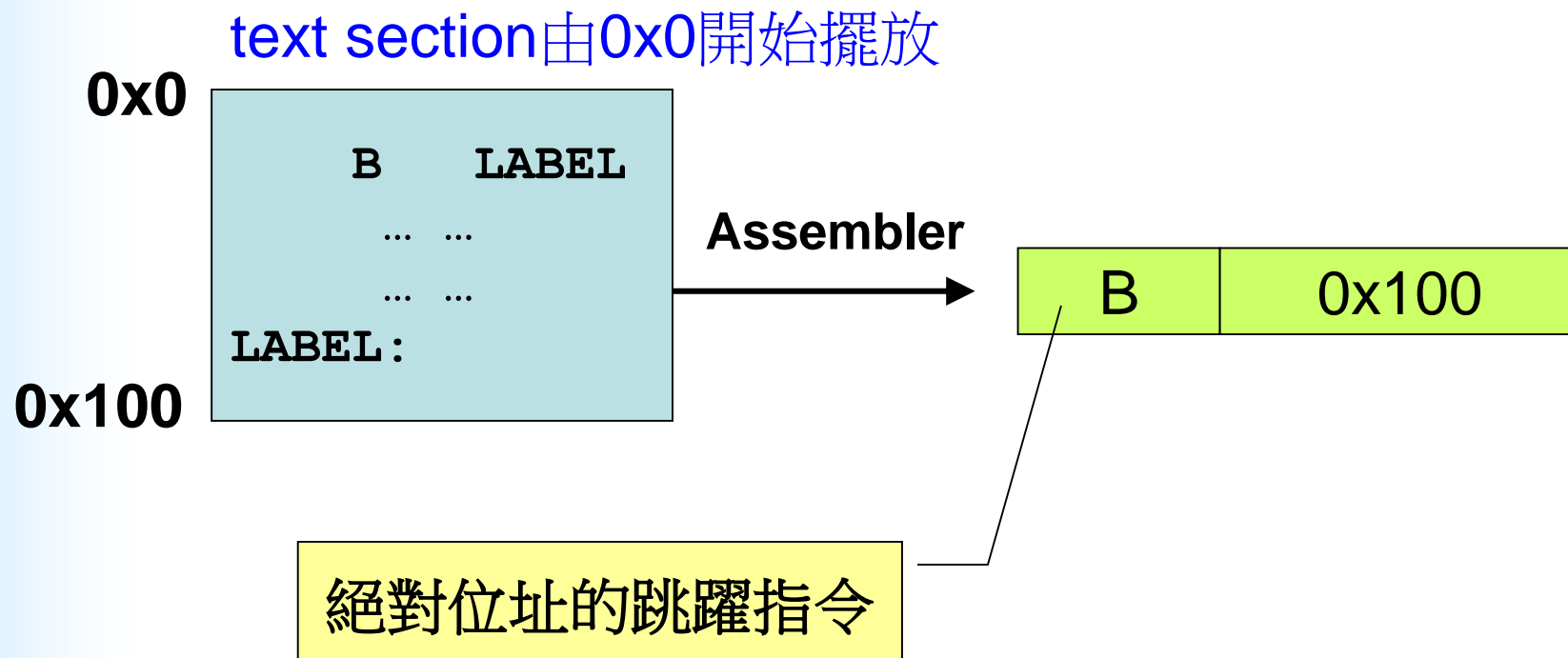
Fall, 2017

# Position Independent Code

- 程式的執行與程式擺放在記憶體的位置無關



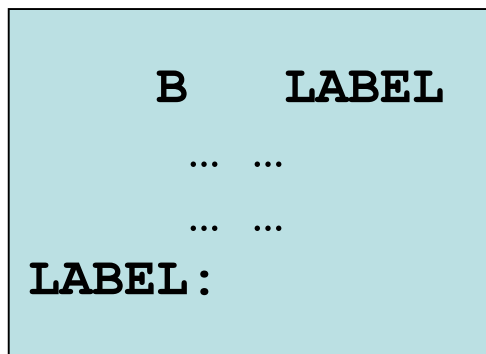
# Example (1)



# Example (2)

程式重新執行，  
text section由0x100開始擺放

0x100



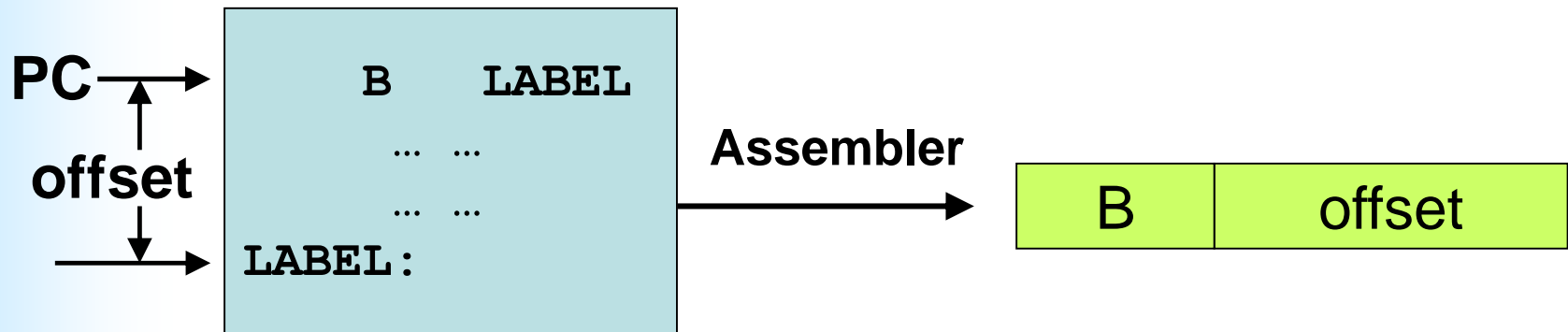
跳躍的位址跟程式由哪裡開始執行有關  
=> **position dependent code**

# Position Independent Code (1)

- 程式的執行跟程式擺放在記憶體哪個位置無關
- 有彈性
- Loader或OS可根據目前系統的狀態，將程式載入到記憶體最適當的地方

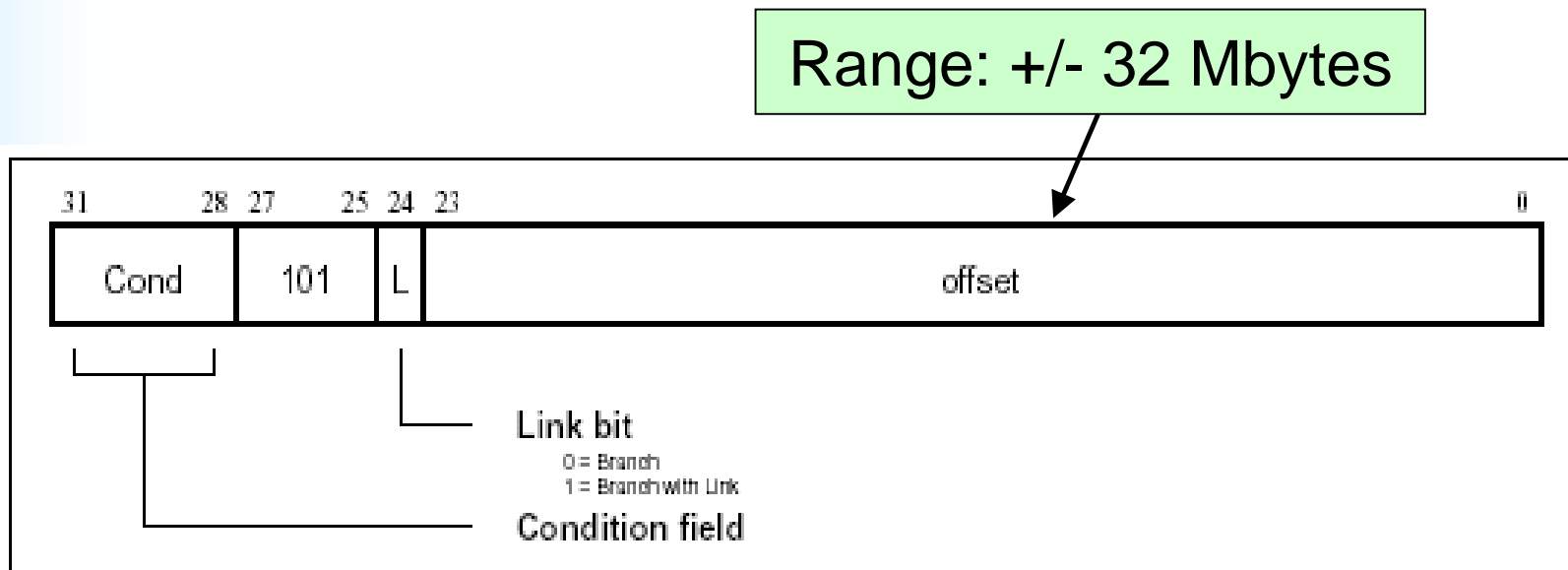
# Position Independent Code (2)

- 使用 **relative** jump instruction
- Ex: PC-relative => 只記錄PC與target位址的差
- Ex: 以某個register為base，紀錄target與該register的差 (offset)



# ARM ISA: Branch and Branch with Link

When the processor executes a branch instruction, **the offset is added to the PC**, and the machine begins fetching instructions from this new address.




Assembler syntax: **B** {L} {cond} <expression>

# Position Independent Code (3)

- ARM: PC-relative
- 若硬體沒有PC-relative jump的指令
  - 自己動手做

Pseudo code

```
                mov r0, #(LABEL - Here)
                B    (pc, r0)
Here:           ???
                ... ..
                ... ..
LABEL:
```

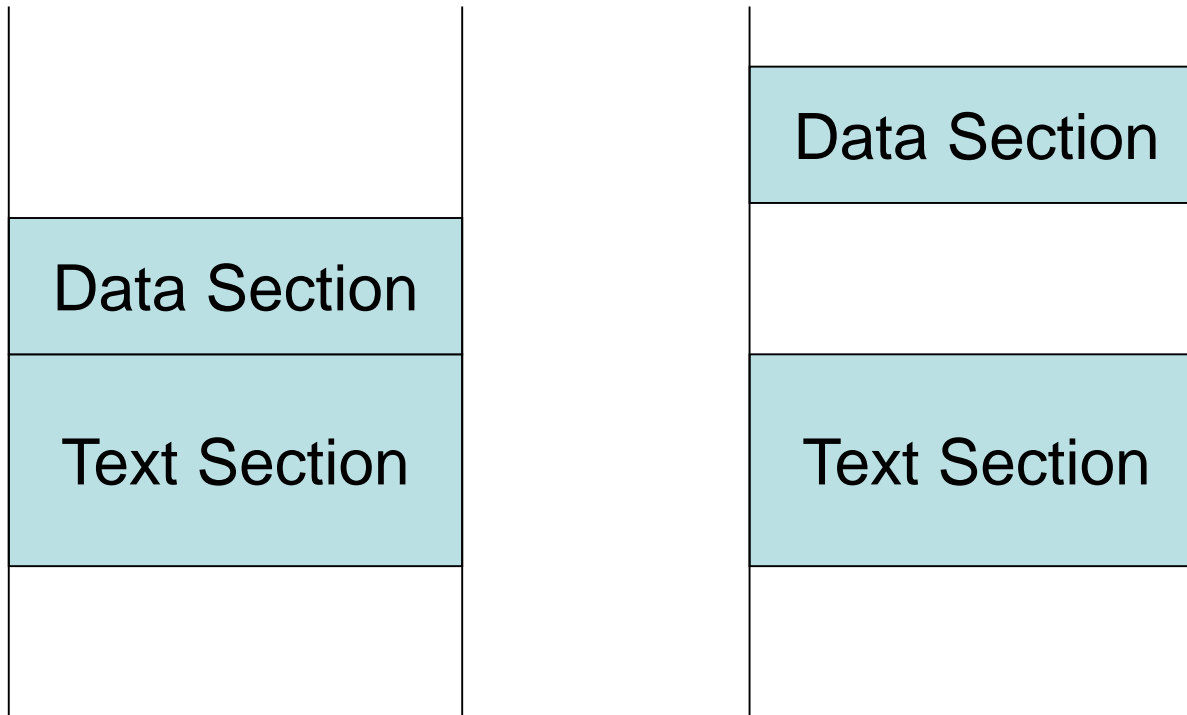


branch to the address of (pc + r0)



# Position Independent Data (1)

- 程式中資料的存取與資料擺放的位置無關



# Position Independent Data (2)

data section由0x0開始擺放

0x10

```
.data  
Here:  
    .word 0x123  
    ...
```

```
ldr r0, [Here]  
...
```

Assembler

ldr	r0	0x10
-----	----	------

# Position Independent Data (3)

data section由0x90開始擺放

0x100

```
.data  
Here:  
    .word 0x123  
    ...
```

0x10 ?

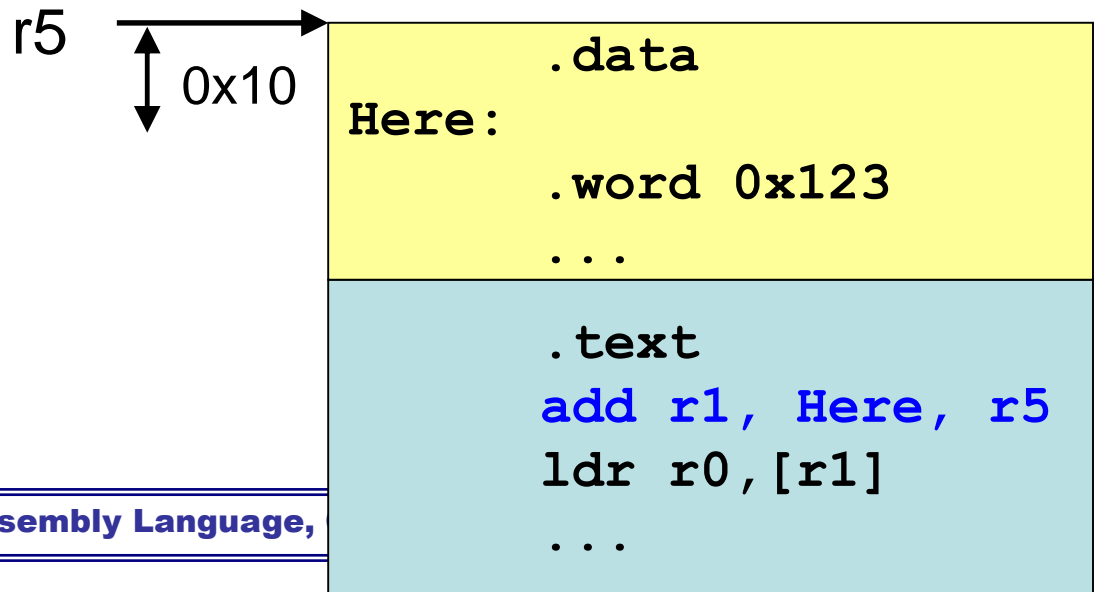
```
.text  
ldr r0, [Here]  
    ...
```

Assembler

ldr	r0	0x10
-----	----	------

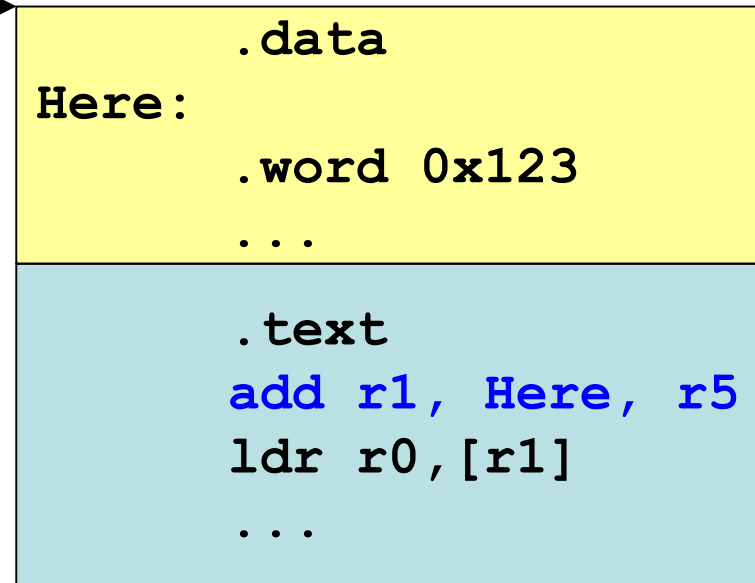
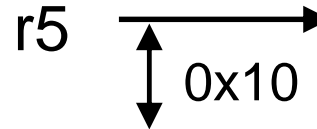
# Position Independent Data (4)

- ARM: *adr* is PC-relative
- 自己動手做
  - Data section總是假設從0x0開始擺放
  - 程式執行時再由OS自由擺放，開始擺放的位址存放在某個register (ex: r5)



# Position Independent Data (5)

- **OS將data section擺在0x0**
  - OS會將0x0指定給r5
  - 程式執行時抓到正確的Here的值  
 $0x10 + 0x0 = 0x10$
- **OS將data section擺在0x90**
  - OS會將0x90指定給r5
  - 程式執行時抓到正確的Here的值  
 $0x10 + 0x90 = 0x100$



# Backup

# Position Independent Code (3)

- ARM: PC-relative
- 若硬體沒有PC-relative jump的指令
  - 自己動手做

```
        mov r0, #(LABEL - Here)
        add r0, r0, 4
        add r0, r0, pc
        B    r0
Here:    ???
        ...
        ...
LABEL:
```