Book：系統程式概論精要 / 賈蓉生，拼大派…

No.

DATE.

系統程式：System Programming.

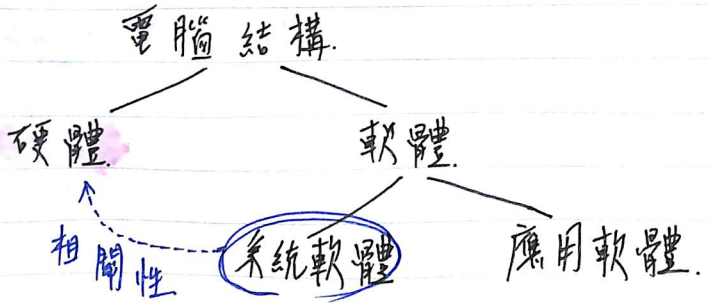
① 系統結構：硬體結構 ↔ 系統軟體.
系統計算架構，機械語言，組合語言

② 系統作業程式：驅動硬體激發組織資源，資料傳輸
i) 行程管理 Process Management：行程 & 執行緒，CPU排程，行程與同步並行
ii) 死結 Deadlocks：資源分配，預防，資源釋放法.
iii) 儲存管理 iv) 分散式系統

③ 系統編譯程式：建立硬體可接受的執行機械碼，
用以執行應用程式之要求功能.
i) 自型分析：編譯模型，語言分析，語法分析.
ii) 語意分析 iii) 編譯碼最佳化

④ 系統資料庫程式：用以執行資料運算、資料存取.（省略）

⑤ 系統組合語言：Assembler Language 格式.




① 系統結構 System Organization
- 系統硬體：電腦實體.
- 系統軟體：使實體有生命活力，能讀取資料、能計算資料，能儲存資料
（電腦能了解機器語言，以 0, 1 了解指令.
人利用組合語言，以文字了解指令.）

- 處理器(Processer)與指令(Instruction)的互動過程.
M：記憶體 Memory > MM, RM, RR 間之互動.
R：暫存器 Register
↳ 資料在記憶體/暫存器間之①移動與②計算

NAN PAO

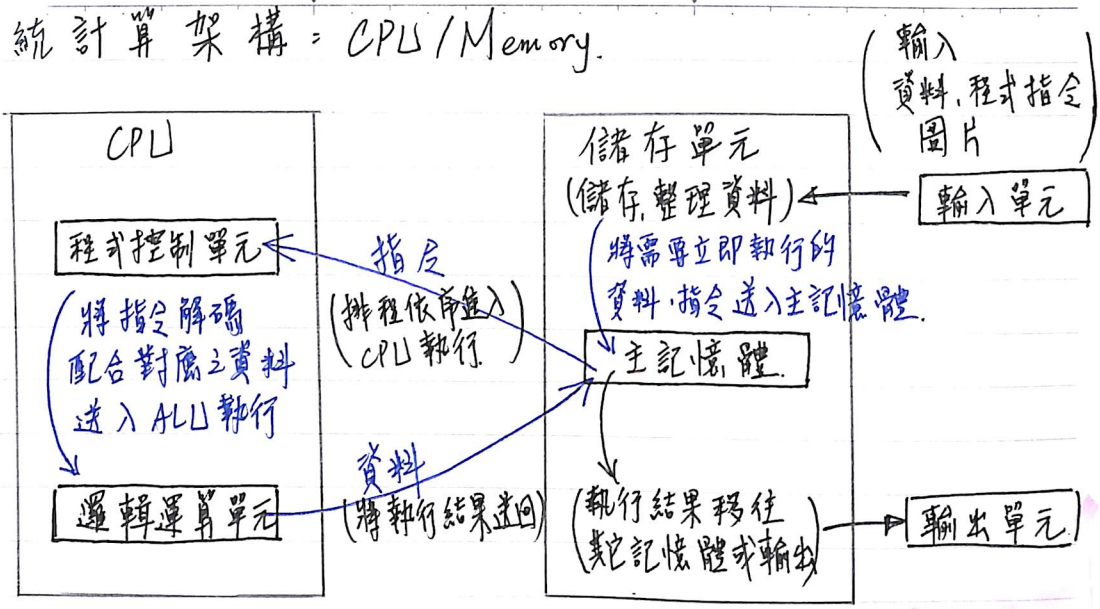
PC - Program Counter.

PCU - Program Control unit 程式控控單元.

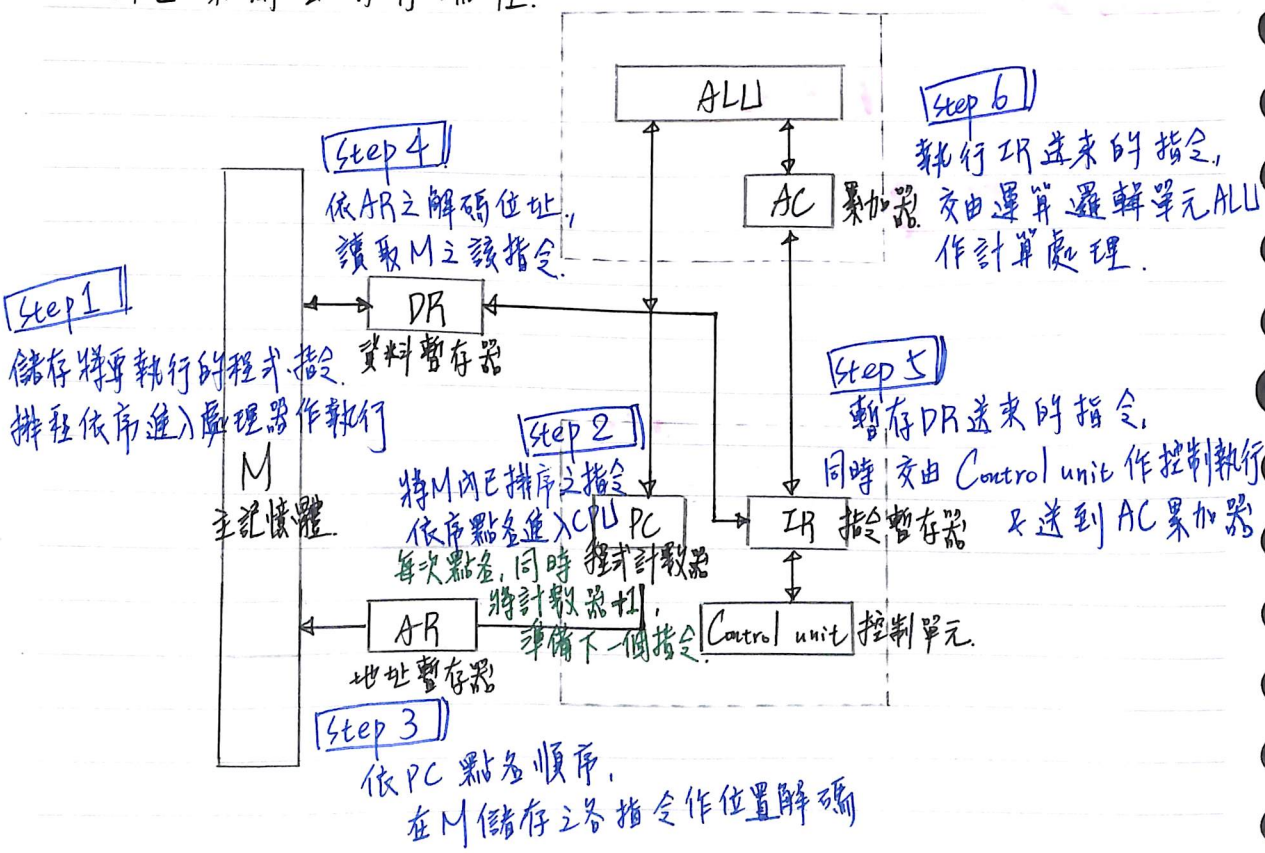
No.

DATE.

- 系統計算架構：CPU / Memory.




- CPU 架構 & 執行流程.



No.

DATE.

Step 6-1:
if 指令"只有操作碼", AC執行IR送來的指令, 並交由ALU作計算處理, 並將執行結果交由DR, 送至M儲存.

Step 6-2:
if 指令有操作碼+資料碼 (ex: add $d_1, d_2$), AC會先暫存操作碼 (add), 另等待資料碼 ($d_1$, $d_2$)
重複 step 2~5, 從M讀取資料碼 ($d_1$, $d_2$),
此時AC有完整的指令 (add $d_1, d_2$),
並交由ALU作計算處理, 並將執行結果交由DR,
送至M儲存.

- 處理器 (Processor) & 指令 (Instruction) 執行.
- Instruction { 操作碼 Opcode / 資料碼 Operand } ex: (低階) 組合語言.

執行
(加法) add $d_1, d_2$ → $d_1 + d_2$.
(停止) stop.
操作碼 資料碼

(0000-0001) (0010-1100)
01 3C ← 資料

- 組合語言: LR $R_3$, $R_5$
18 3 5 ← 指令操作碼 18 35
(0001-1000-0011-0101)

將暫存器 $R_5$ 的內容,
載入另一個暫存器 $R_3$,
並取代原有內容.

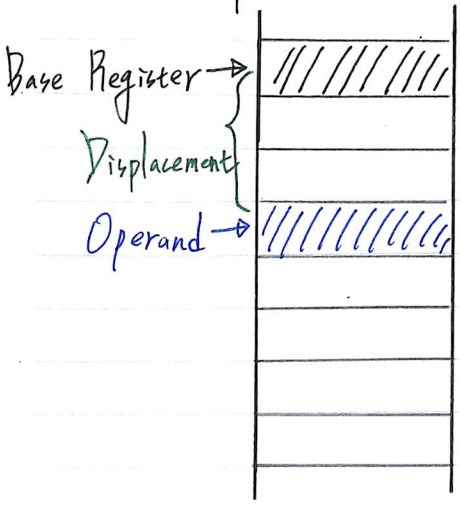
- 組合語言: 為 機械語言 的符號.
文字字串 對應之數字字串.

No.

DATE.

- 指令 (組合語言)
  - 操作碼 Opcode
    - 指令操作功能
      - 計算指令：經控制單元，計算單元，執行計算與邏輯操作
      - 輸入/輸出指令：資料 輸出/入.
      - 搬移指令：將資料從原儲存位置 搬移至另一個儲存位置.
      - 控制指令：控制執行流程.
  - 資料碼 Operand.
    - 配合操作碼之要求，讀取儲存暫存器或主記憶體內之資料.
      - 暫存器資料：存取方式 ex: LR R3, R5.
      - 記憶體資料 →

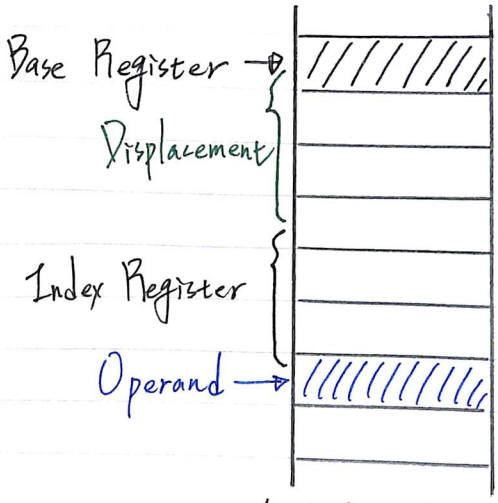
- 記憶體資料：
  - 暫存器 Base　儲存基址.
  - 暫存器 Index　儲存索引址.
  - Displacement　間址.




無 Index Register 作用

資料位址 = 基址 + 間址.

Effective Address




有 Index Register 作用.

資料位址 = 基址 + 間址 + 索引值.

ex: 指令機械碼 5f389018，其對應組合語言為：

載入暫存器　Displacement 的十進位　Index Register.　Base Register.

L　R3, 24(R8, R9)

操作碼=5f　資料碼

功能：將主記憶體某個地址 (24(R8, R9)) 之內容，載入一個指定暫存器 (R3)

假設　R8 之內容　00009834

R9 之內容　00000014

則 Effective Address 為：

00009834 ← R8 Index

00000014 ← R9 Base.

+) 018 ← Displacement

00009860

↑超過15，歸0進位.

假設此時主記憶體地址 00009860 之內容是 0002，則將被載入 R3，∴暫存器 R3 之內容為 0002.

② 系統作業程式 System Operation. Programming.

- 排程 Process Scheduling

循環分時排程，有意義的排序

多個行程等待，為了有效執行，系統程式將依其迫切性排序等待執行 (就緒佇列 Ready Queues)

- 死結 Dead Lock

資源的不數使用

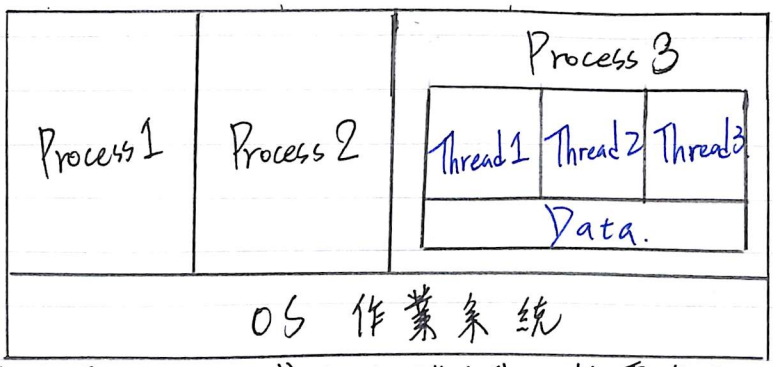
在多工環境中，隨時都有多個行程 (Processes) 或執行緒 (Threads) 競爭有限數量的資源 (Resources)

No.

DATE.

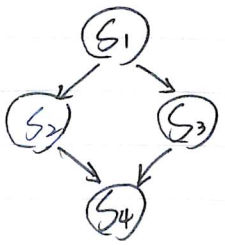
- 行程 Process
  指正在被執行的程式，是個工作單元，系統內有多個行程同時存在，但 CPU 一個時間只處理一個行程，其多數的行程皆為等待狀態 Waiting State.

- 執行緒 Thread.
  為了使用執行效果生動自然，付予某些行程競爭特質，競爭搶入 CPU 執行
  一個多元性工作的行程 Process，
  可依其不同工作內容分割出多個執行緒 Thread.
  優點：工作分擔，省時省資源




- 當一個行程(Process)或執行緒(Thread)要執行一個工作(Job)時，通常需要抓取某些資源(Resoures)伴隨執行，
  其中 3 個過程 Sequence：
  ① 要求 Request
  ② 使用 Use
  ③ 釋放 Release
  → 皆依賴系統程式呼叫 System Call 執行

- 在多個行程同步並行時，為保持資料一致性，
  設有程式片斷：
  $S_1$ : $a \leftarrow 5$
  $S_2$ : $b \leftarrow a+1$
  $S_3$ : $c \leftarrow a+2$
  $S_4$ : $d \leftarrow b+c$



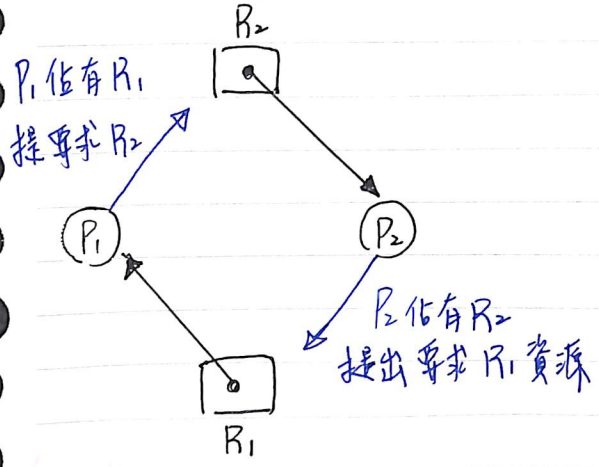

NAN PAO

No.

DATE.

☆ 死結 DeadLock

循環等待 Circular Wait (永久等待，無法解開的迴圈)




行程集合 $P=\{P_1, P_2\}$

資源集合 $R=\{R_1, R_2\}$

關係邊集合

$E=\{R_1 \to P_1 \to R_2 \to P_2 \to R_1\}$.

- 消弭死結方法 〈行程終止法 / 資源釋放法〉

飢餓現象 Starvation

使用時間戳記 (TS Timestamp).

- 記憶體地址 Memory Address (兩種表示方式)

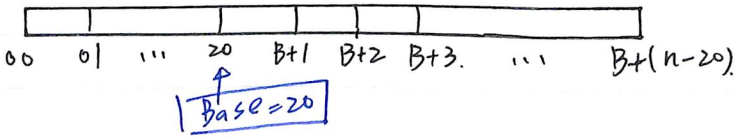
① 絕對地址 (Absolute Address / Physical Address)

設定記憶體起始位置為「0」


② 相對地址 (Relative Address / Logical Address)

設定一地址基點 Base，各地址從基點依序計算，

可視為重新定址 (Relocaation)




- 絕對地址 = 基點 + 相對地址.

No.

DATE.

## ③ 系統編譯程式 System Compiling Programming.

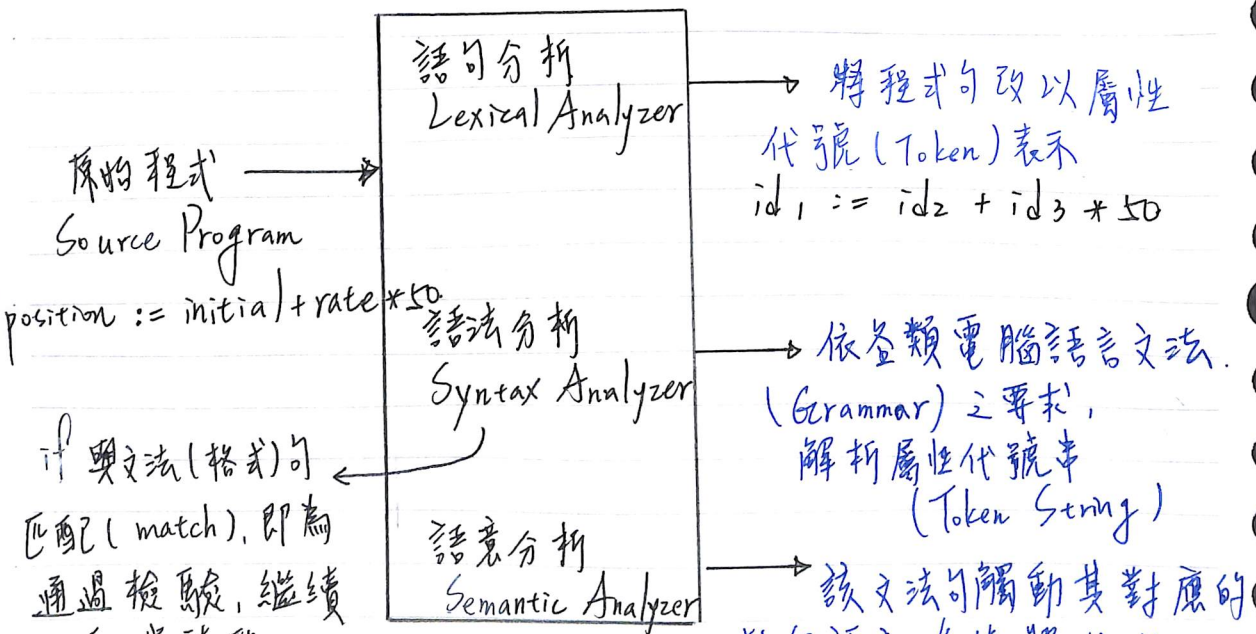
原始程式 Source Program → 編譯 → (機械型式的) 目的程式 Target Program.

ex: 組合語言程式.

編譯過程:
- 語句分析
- 語法分析
- 語意分析

Compiler.


- 編譯模型: 句型分析




if 與文法(格式)句匹配(match),即為通過檢驗,繼續不一步流程

if 有文法錯誤(Syntax Error),表即原始程式有錯誤.

No.

DATE.

position := initial + rate * 50

(程式句)

**語句分析**

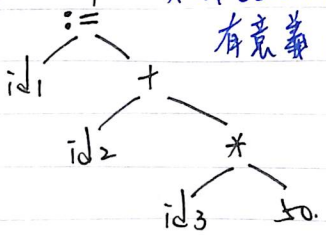
檢驗原始程式 Source Program 是否能被電腦系統接受.

$id_1 := id_2 + id_3 * 50$

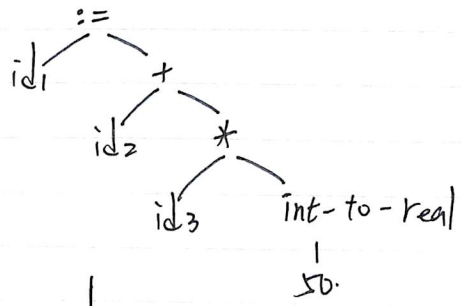
(屬性代號)

**語法分析**

文法 Grammar 有意義的表達




**語意分析**




編譯碼產生器 Code Generter.

**句型分析流程**

句型分析 Sentence Analyzer.

**臨時碼產生器**

Intermediate Code Generator

$temp_1 := int\text{-}to\text{-}real\ (50)$

$temp_2 := id_3 * temp_1$

$temp_3 := id_2 + temp_2$

$id_1 := temp_3$.

**最佳化編譯器**

Code Optimizer

$temp_1 := id_3 * 50.0$

$id_1 := id_2 + temp_1$.

**編譯碼產生器**

Code Generator

```
mov   R2, id3
mult  R2, #50.0
mov   R1, id2
add   R1, R2
mov   id1, R1
```

**編譯碼產生流程**

NAN PAO

No.

DATE.

## ⑤ 系統組合語言程式 Assembler Language Programming.

- 組合語言格式：

以"列(Line)"為敘述句(Statement)，
每一敘述佔用一列，每列又分4個區(Field)，
區間以空格(Space)分隔

- 名稱區 Name Field

名稱(Name)或位置符號(Location) → 用於 { 指令 Instruction / 常數 Constant / 資料區 Data Area / 定義 Definition }

→ 唯有名稱由使用者撰寫：

(1) 名稱長度不得多於8個符號
(2) 名稱第一個符號為字母，後者才可用字母or數字
(3) 字母A-Z，數字0-9，常用其他符號$，@，#，_

ex: A1，PAIB，#0ZG38U —— (O)
3B，A1BC3D567，Z*3. —— (X)
↑ 運算符號

- 操作指令區 Operation Field.

用於指令符號 { 機器碼指令 / 常數或符號定義 / 組合語言指令 / 巨集指令 }

- 應用資料區 Operand Field.

用於一個或多個資料 Operands.
資料與資料間以逗號(,)分隔(不得用空格)

- 註解區 Remark Field

僅供解說，內容不參與執行.