開放電腦計劃

報告人：陳鍾誠

2011 年 8 月 20 日

台灣開源人年會 - 中研院

開放電腦計畫 -- 動機

- 學術界的一個重要任務
  - 將工業技術研究清楚，並且寫成書籍或技術報告。
  - 讓學生能習得重要的技術，理解工業結構與原理。

開放電腦計畫 -- 原則

- 採用 KISS (Keep It Simple and Stupid) 原則，越簡單越好。
- 不需要創新，盡可能採用成熟的舊技術。
- 重視原理、文件、而非程式技術

開放電腦計畫 -- 目標

- 設計一台完整的「開放原始碼電腦」，從軟體到硬體通通都是開放原始碼的。
- CPU 、主機板、虛擬機、組譯器、編譯器、連結器、嵌入式作業系統、作業系統

開放電腦計畫 -- 子專案

元件	實現方式	設計方式	專案成品	負責人	狀態
CPU 硬體	FPGA+VHDL	自行撰寫	CPU1	林宗憲	參考 RichardCPU1, 研究中
BUS+I/O	電路板	修改自 Arduino	BOARD1	專題生	研究修改中
虛擬機	C 語言	自行撰寫	VM1	陳鍾誠	已完成
組譯器	C 語言	自行撰寫	AS1	陳鍾誠	已完成
連結器	C 語言	自行撰寫	LD1	陳鍾誠	參考 Jserv 的 ndl
編譯器	C 語言	自行撰寫	CC1	陳鍾誠	參考 TinyCC, 與 Plan9 的 ken-cc
嵌入式作業系統	C 語言	自行撰寫	EOS1	陳鍾誠	研究修改中
作業系統	C 語言	修改自 UNIXxv6	OS1	陳鍾誠	研究修改中

開放電腦計畫 -- 基礎

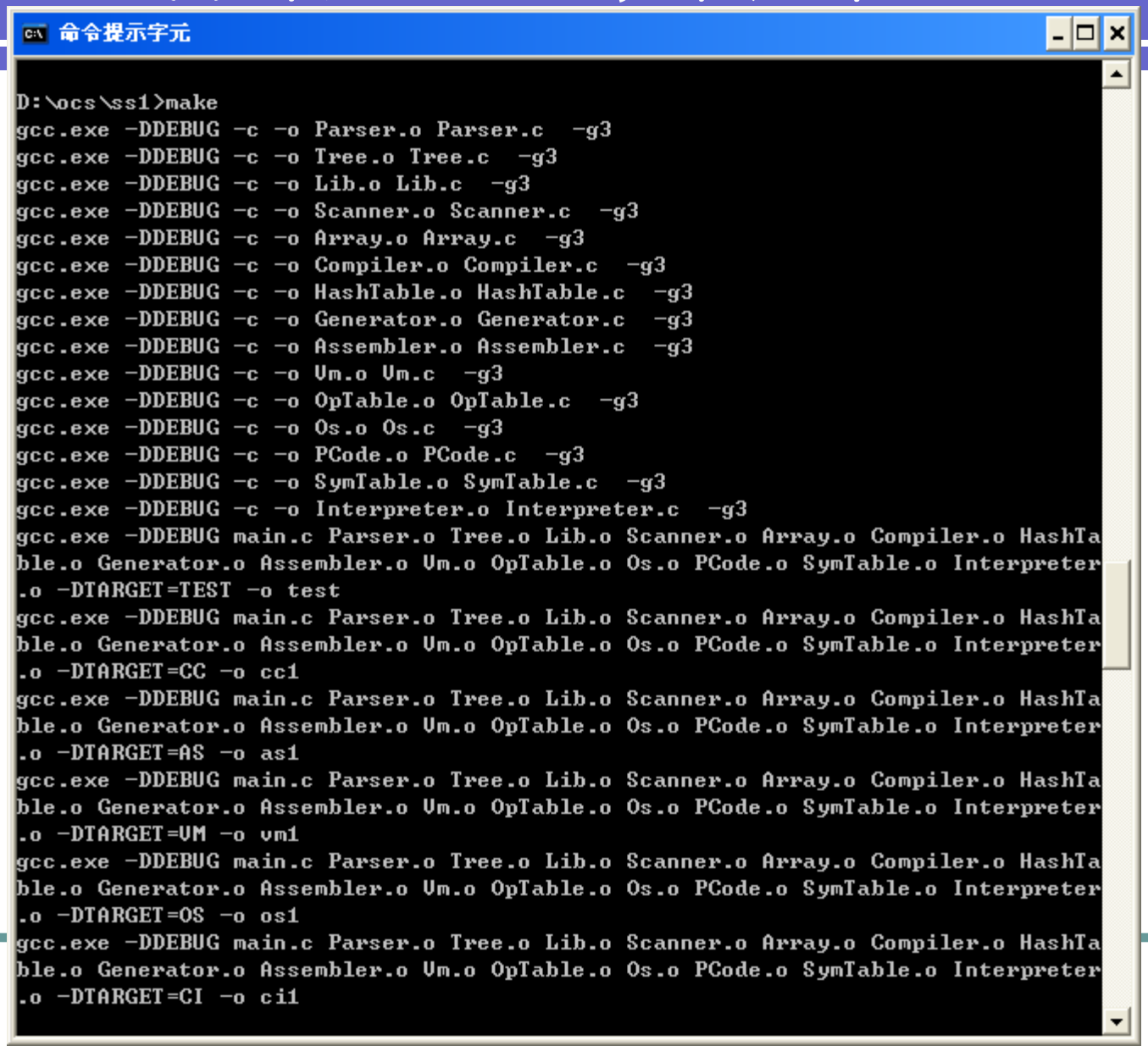
- 2010 年出版了系統程式一書
  - 定義了
    - CPU0： 處理器
  - 實作了
    - VM0： 虛擬機
    - AS0： 組譯器
    - C0C： 編譯器

第 1 章 系統軟體
第 2 章 電腦的硬體結構
第 3 章 組合語言
第 4 章 組譯器
第 5 章 連結與載入
第 6 章 巨集處理器
第 7 章 高階語言
第 8 章 編譯器
第 9 章 虛擬機器
第 10 章 作業系統
第 11 章 嵌入式系統
第 12 章 系統軟體實作

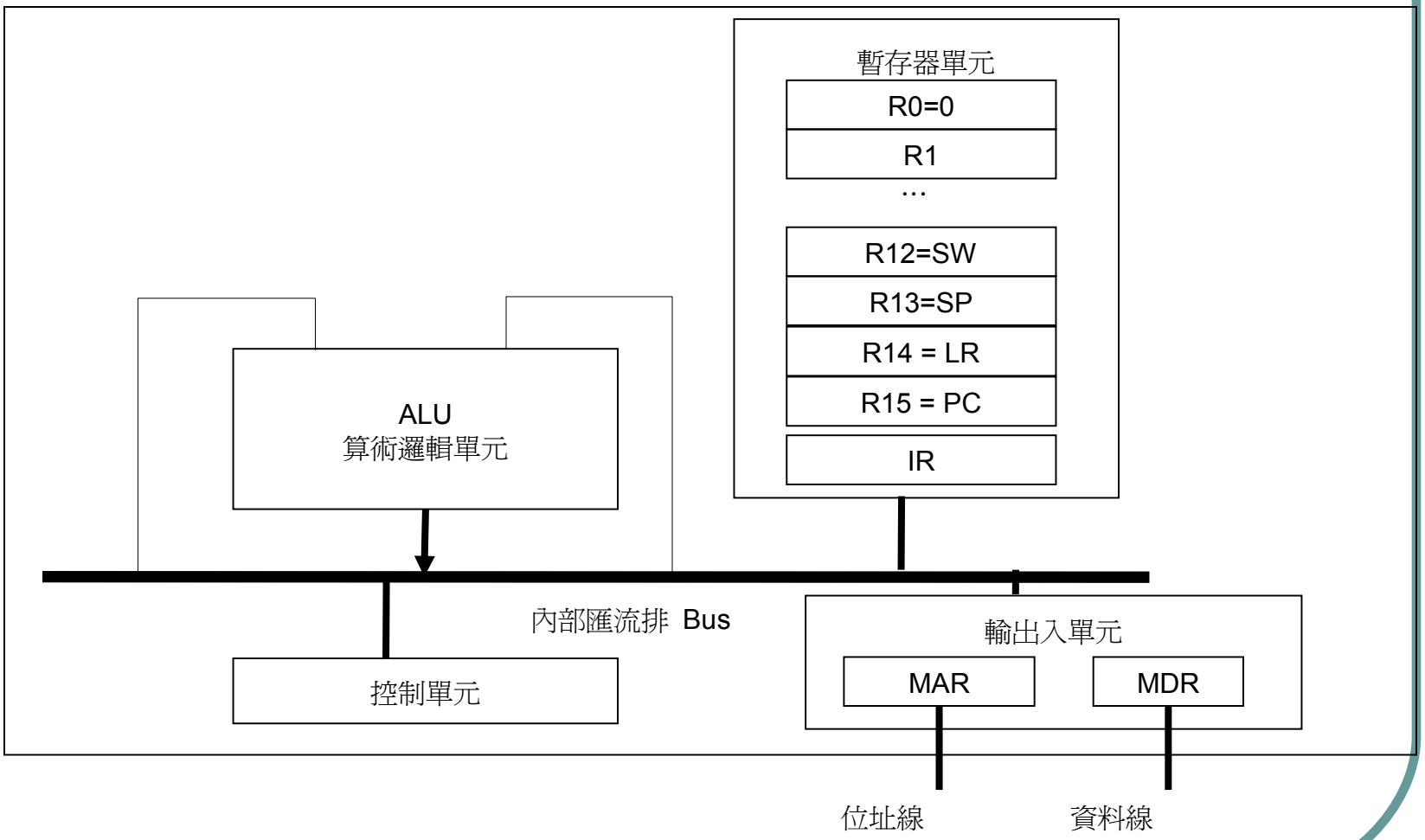
開放電腦計畫 – 現況

- 處理器
  - CPU0 → CPU1 規格確定
  - CPU1 Verilog 實作　　研究中
- 虛擬機
  - VM0 → VM1 已完成
- 組譯器
  - AS0 → AS1 已完成
- 連結器：
  - 尚未開始，目前先將全部程式連接起來組譯，暫時不需連結。
- 編譯器
  - C0C → CC1
    - 已完成：剖析器、語意分析、符號表
    - 撰寫中：程式碼產生
- 作業系統
  - eos0: 已完成
    - 兩個行程切換 (74 行 CPU1 的組合語言)
  - eos1: 尚未開始
  - os1: 尚未開始

開放電腦計畫 - 專案建置


命令提示字元
D:\ocs\ss1>make
gcc.exe -DDEBUG -c -o Parser.o Parser.c -g3
gcc.exe -DDEBUG -c -o Tree.o Tree.c -g3
gcc.exe -DDEBUG -c -o Lib.o Lib.c -g3
gcc.exe -DDEBUG -c -o Scanner.o Scanner.c -g3
gcc.exe -DDEBUG -c -o Array.o Array.c -g3
gcc.exe -DDEBUG -c -o Compiler.o Compiler.c -g3
gcc.exe -DDEBUG -c -o HashTable.o HashTable.c -g3
gcc.exe -DDEBUG -c -o Generator.o Generator.c -g3
gcc.exe -DDEBUG -c -o Assembler.o Assembler.c -g3
gcc.exe -DDEBUG -c -o Vm.o Vm.c -g3
gcc.exe -DDEBUG -c -o OpTable.o OpTable.c -g3
gcc.exe -DDEBUG -c -o Os.o Os.c -g3
gcc.exe -DDEBUG -c -o PCode.o PCode.c -g3
gcc.exe -DDEBUG -c -o SymTable.o SymTable.c -g3
gcc.exe -DDEBUG -c -o Interpreter.o Interpreter.c -g3
gcc.exe -DDEBUG main.c Parser.o Tree.o Lib.o Scanner.o Array.o Compiler.o HashTable.o Generator.o Assembler.o Vm.o OpTable.o Os.o PCode.o SymTable.o Interpreter.o -DTARGET=TEST -o test
gcc.exe -DDEBUG main.c Parser.o Tree.o Lib.o Scanner.o Array.o Compiler.o HashTable.o Generator.o Assembler.o Vm.o OpTable.o Os.o PCode.o SymTable.o Interpreter.o -DTARGET=CC -o cc1
gcc.exe -DDEBUG main.c Parser.o Tree.o Lib.o Scanner.o Array.o Compiler.o HashTable.o Generator.o Assembler.o Vm.o OpTable.o Os.o PCode.o SymTable.o Interpreter.o -DTARGET=AS -o as1
gcc.exe -DDEBUG main.c Parser.o Tree.o Lib.o Scanner.o Array.o Compiler.o HashTable.o Generator.o Assembler.o Vm.o OpTable.o Os.o PCode.o SymTable.o Interpreter.o -DTARGET=VM -o vm1
gcc.exe -DDEBUG main.c Parser.o Tree.o Lib.o Scanner.o Array.o Compiler.o HashTable.o Generator.o Assembler.o Vm.o OpTable.o Os.o PCode.o SymTable.o Interpreter.o -DTARGET=OS -o os1
gcc.exe -DDEBUG main.c Parser.o Tree.o Lib.o Scanner.o Array.o Compiler.o HashTable.o Generator.o Assembler.o Vm.o OpTable.o Os.o PCode.o SymTable.o Interpreter.o -DTARGET=CI -o ci1

開放電腦計畫 – CPU1


暫存器單元
R0=0
R1
…
R12=SW
R13=SP
R14 = LR
R15 = PC
IR
ALU
算術邏輯單元
內部匯流排 Bus
控制單元
輸出入單元
MAR
MDR
位址線
資料線

CPU1 -- 暫存器

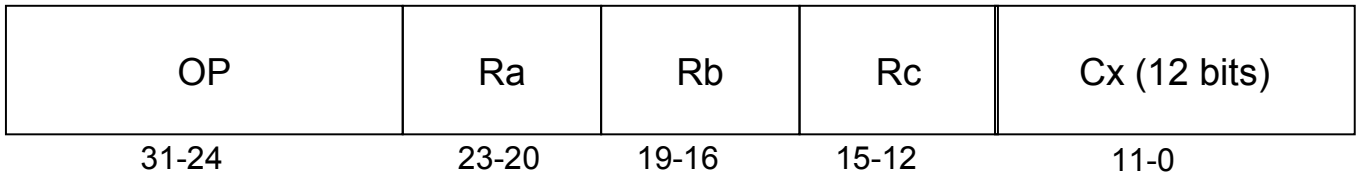
- R0
  - 唯讀暫存器。 R0 的值永遠都是常數零。
- R1-R15
  - 15 個可存取暫存器
  - R12 ：狀態暫存器 (Status Word, SW)
  - R13 ：堆疊暫存器 (Stack Pointer Register, SP)
  - R14 ：連結暫存器 (Link Register, LR)
  - R15 ：程式計數器 (Program Counter, PC)

CPU1 – 指令集

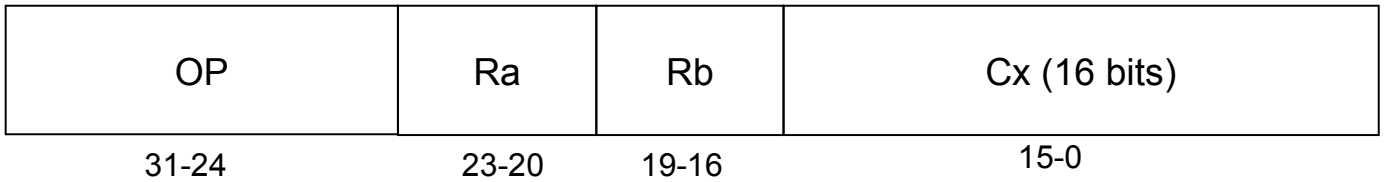
- 包含四大類的指令
  - 載入儲存 : LD, ST, …
  - 運算指令 : ADD, SUB, XOR, SHL, ROL, …
  - 跳躍指令 : JMP, JGT, JGE, CALL, INT
  - 堆疊指令 : PUSH, POP, …

CPU1 – 指令格式

A 型



L 型



J 型

CPU1 – 載入儲存指令

類型	格式	指令	OP	說明	語法	語義
載入儲存	L	LD[2]	00	載入 word	LD Ra, [Rb+Cx]	Ra←[Rb+ Cx]
	L	ST	01	儲存 word	ST Ra, [Rb+ Cx]	Ra→[Rb+ Cx]
	L	LDB	02	載入 byte	LDB Ra, [Rb+ Cx]	Ra←(byte)[Rb+ Cx]
	L	STB	03	儲存 byte	STB Ra, [Rb+ Cx]	Ra→(byte)[Rb+ Cx]
	A	LDR	04	LD (暫存器版)	LDR Ra, [Rb+Rc]	Ra→(byte)[Rb+ Rc]
	A	STR	05	ST (暫存器版)	STR Ra, [Rb+Rc]	Ra→[Rb+ Rc]
	A	LBR	06	LDB (暫存器版)	LBR Ra, [Rb+Rc]	Ra←(byte)[Rb+ Rc]
	A	SBR	07	STB (暫存器版)	SBR Ra, [Rb+Rc]	Ra→(byte)[Rb+ Rc]
	L	LDI	08	立即載入	LDI Ra, Rb+Cx	Ra← Rb + Cx

CPU1 – 運算指令

類型	格式	指令	OP	說明	語法	語義
運算指令	A	CMP[3]	10	比較	CMP Ra, Rb	SW ← Ra >=< Rb
	A	MOV	12	移動	MOV Ra, Rb	Ra ← Rb
	A	ADD	13	加法	ADD Ra, Rb, Rc	Ra ← Rb+Rc
	A	SUB	14	減法	SUB Ra, Rb, Rc	Ra ← Rb-Rc
	A	MUL	15	乘法	MUL Ra, Rb, Rc	Ra ← Rb*Rc
	A	DIV	16	除法	DIV Ra, Rb, Rc	Ra ← Rb/Rc
	A	AND	18	邏輯 AND	AND Ra, Rb, Rc	Ra ← Rb and Rc
	A	OR	19	邏輯 OR	OR Ra, Rb, Rc	Ra ← Rb or Rc
	A	XOR	1A	邏輯 XOR	XOR Ra, Rb, Rc	Ra ← Rb xor Rc
	A	ROL[4]	1C	向左旋轉	ROL Ra, Rb, Cx	Ra ← Rb rol Cx
	A	ROR	1D	向右旋轉	ROR Ra, Rb, Cx	Ra ← Rb ror Cx
	A	SHL	1E	向左移位	SHL Ra, Rb, Cx	Ra ← Rb << Cx
	A	SHR	1F	向右移位	SHR Ra, Rb, Cx	Ra ← Rb >> Cx

CPU1 – 跳躍指令

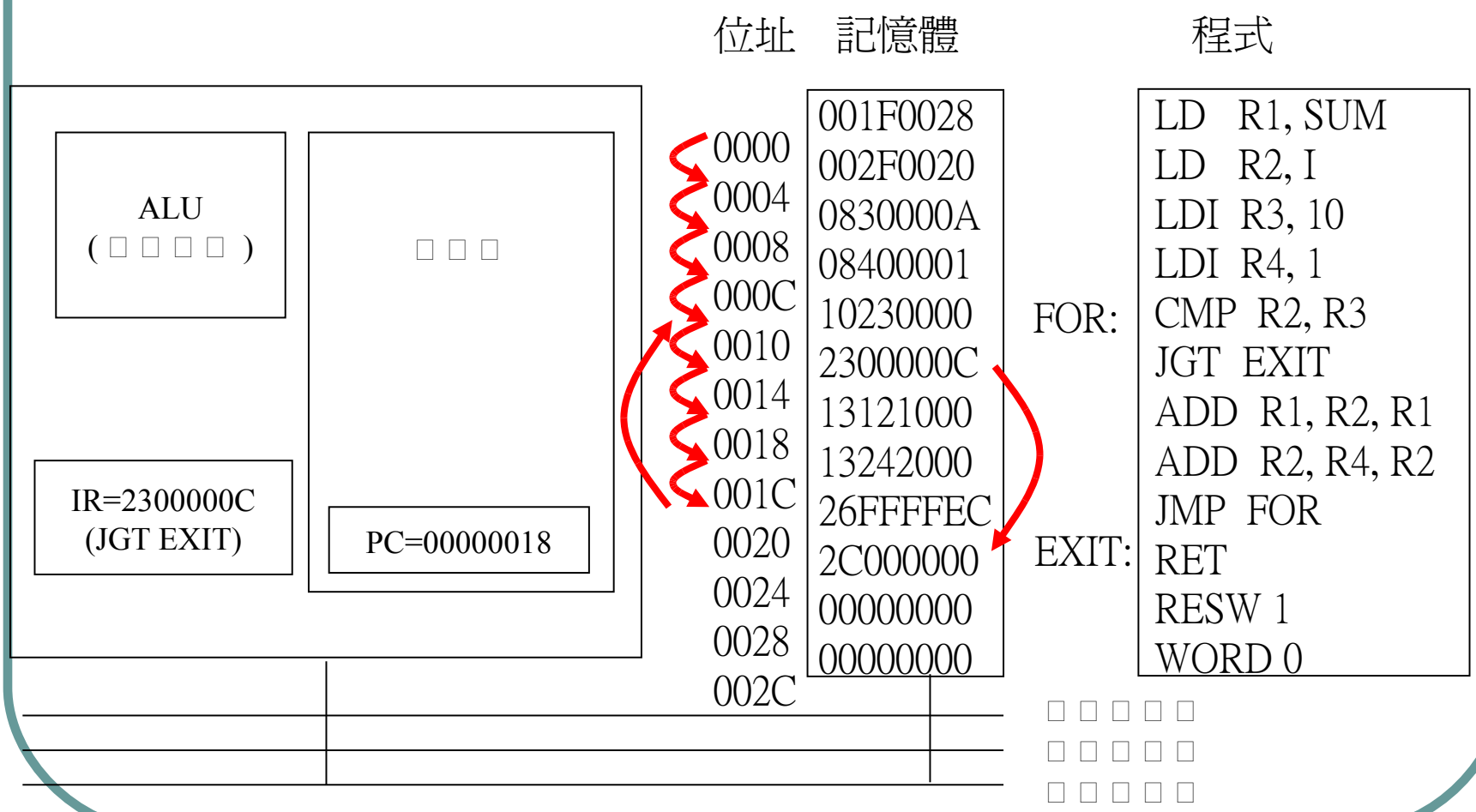
類型	格式	指令	OP	說明	語法	語義
跳躍指令	J	JEQ[5]	20	跳躍 (相等)	JEQ Cx	if SW(=) PC ← PC+Cx
	J	JNE	21	跳躍 (不相等)	JNE Cx	if SW(!=) PC ← PC+Cx
	J	JLT	22	跳躍 (<)	JLT Cx	if SW(<) PC ← PC+Cx
跳躍指令	J	JGT	23	跳躍 (>)	JGT Cx	If SW(>) PC ← PC+Cx
	J	JLE	24	跳躍 (<=)	JLE Cx	if SW(<=) PC ← PC+Cx
	J	JGE	25	跳躍 (>=)	JGE Cx	If SW(>=) PC ← PC+Cx
	J	JMP	26	跳躍 (無條件)	JMP Cx	PC ← PC+Cx
	J	SWI[6]	2A	軟體中斷	SWI Cx	LR ← PC; PC ← Cx
	J	CALL	2B	跳到副程式	CALL Cx	LR← PC; PC ← PC+Cx
	J	RET	2C	返回	RET	PC ← LR

CPU1 – 堆疊指令

類型	格式	指令	OP	說明	語法	語義
堆疊指令	A	PUSH	30	推入 word	PUSH Ra	SP-=4; [SP] = Ra;
	A	POP	31	彈出 word	POP Ra	Ra = [SP]; SP+=4;
	A	PUSHB	32	推入 byte	PUSHB Ra	SP--; [SP] = Ra; (byte)
	A	POPB	33	彈出 byte	POPB Ra	Ra = [SP]; SP++; (byte)

CPU1 – 指令集

CPU1 – 程式、編碼與執行



CPU1 的組合語言範例

- sum = 1+2+….+10

```
        LD      R1, sum
        LD      R2, i
        LDI     R3, 10
        LDI     R4, 1
FOR:    CMP     R2, R3
        JGT     EXIT
        ADD     R1, R2, R1
        ADD     R2, R4, R2
        JMP     FOR
EXIT:   RET
i:      RESW    1
sum:    WORD    0
```

開放電腦計畫 – AS1 組譯器

```
D:\ocs\ss1>as1 sum.asm1 sum.obj1
Assembler:asmFile=sum.asm1 objFile=sum.obj1
...
=================PASS1================
   0 LD R1, sum                                L  0 (null)
   4 LD R2, i                                  L  0 (null)
   8 LDI R3, 10                                L  8 (null)
...
===============SYMBOL TABLE=========
  10 FOR: CMP R2, R3                           A 10 (null)
  2c sum: WORD 0                               D f2 (null)
  28 i:  RESW 1                                D f0 (null)
  24 EXIT: RET                                 J 2c (null)
=============PASS2==============
   0 LD R1, sum                                L  0 001F0028
   4 LD R2, i                                  L  0 002F0020
   8 LDI R3, 10                                L  8 0830000A
   c LDI R4, 1                                 L  8 08400001
  10 FOR: CMP R2, R3                           A 10 10230000
  14 JGT EXIT                                  J 23 2300000C
  18 ADD R1, R2, R1                            A 13 13121000
  1c ADD R2, R4, R2                            A 13 13242000
  20 JMP FOR                                   J 26 26FFFFEC
  24 EXIT: RET                                 J 2c 2C000000
  28 i:  RESW 1                                D f0 00000000
  2c sum: WORD 0                               D f2 00000000
==========Save to ObjFile:sum.obj1==========
001F0028002F002008300000A0840000110230000230000C13121000132
4200026FFFFEC2C000000000000000000000000
```

開放電腦計畫 - VM1 虛擬機

```
D:\ocs\ss1>vm1 sum.obj1
===Vm:run sum.obj1===
PC=00000000 LR=ffffffff IR=001f0028 SW=00000000 R[01]=0x00000000=0 count=0
PC=00000004 LR=ffffffff IR=002f0020 SW=00000000 R[02]=0x00000000=0 count=0
PC=00000008 LR=ffffffff IR=0830000a SW=00000000 R[03]=0x0000000a=10 count=0
PC=0000000c LR=ffffffff IR=08400001 SW=00000000 R[04]=0x00000001=1 count=0
PC=00000010 LR=ffffffff IR=10230000 SW=80000000 R[12]=0x80000000=-2147483648 count=0
PC=00000014 LR=ffffffff IR=2300000c SW=80000000 R[00]=0x00000000=0 count=0
PC=00000018 LR=ffffffff IR=13121000 SW=80000000 R[01]=0x00000000=0 count=0
...
PC=00000018 LR=ffffffff IR=13121000 SW=80000000 R[01]=0x0000002d=45 count=0
PC=0000001c LR=ffffffff IR=13242000 SW=80000000 R[02]=0x0000000a=10 count=0
PC=00000020 LR=ffffffff IR=26ffffec SW=80000000 R[15]=0x00000010=16 count=0
PC=00000010 LR=ffffffff IR=10230000 SW=40000000 R[12]=0x40000000=1073741824 count=0
PC=00000014 LR=ffffffff IR=2300000c SW=40000000 R[00]=0x00000000=0 count=0
PC=00000018 LR=ffffffff IR=13121000 SW=40000000 R[01]=0x00000037=55 count=0
PC=0000001c LR=ffffffff IR=13242000 SW=40000000 R[02]=0x0000000b=11 count=0
PC=00000020 LR=ffffffff IR=26ffffec SW=40000000 R[15]=0x00000010=16 count=0
PC=00000010 LR=ffffffff IR=10230000 SW=00000000 R[12]=0x00000000=0 count=0
PC=00000014 LR=ffffffff IR=2300000c SW=00000000 R[00]=0x00000000=0 count=0
PC=00000024 LR=ffffffff IR=2c000000 SW=00000000 R[00]=0x00000000=0 count=0

===CPU0 dump registers===
IR =0x2c000000=738197504
R[00]=0x00000000=0 R[01]=0x00000037=55 R[02]=0x0000000b=11 R[03]=0x0000000a=10
R[04]=0x00000001=1 R[05]=0x00000000=0 R[06]=0x00000000=0 R[07]=0x00000000=0
R[08]=0x00000000=0 R[09]=0x00000000=0 R[10]=0x00000000=0 R[11]=0x00000000=0
R[12]=0x00000000=0 R[13]=0x00000000=0 R[14]=0xffffffff=-1 R[15]=0x00000028=40
```

開放電腦計畫 – EOS0 最小作業系統

EOS0 – 兩行程不斷切換

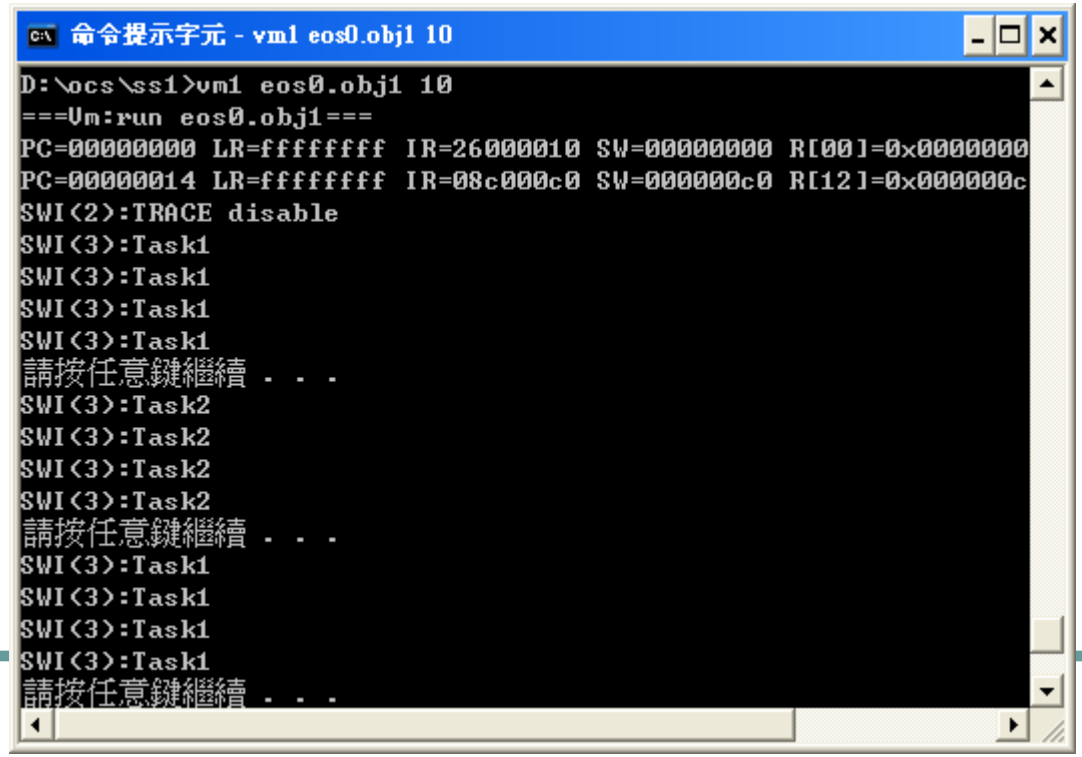
```
1  InterruptVector:
2      JMP ResetHandler
3      JMP Unexpected
4      JMP Unexpected
5      JMP IrqHandler
6  Unexpected:
7      JMP Unexpected
8  ResetHandler:
9      LDI R12, 192          ; 允許軟硬體中斷 (T:SWI)
10     SWI 2                 ; 停止追蹤
11     JMP Task1             ; 啓動時直接執行第一個行程
12 IrqHandler:
13     LDI R12, 64           ; 禁止硬體中斷 (I:IRQ)，允許
14 SaveTask:                 ; 保存目前行程的暫存器到 Ta
15     ST  R14, TR14
16     CALL GetTaskBuf
17     ST  R9, [R1+36]
18     LD  R14, TR14
19     ST  R14, [R1+60]      ; 存入舊行程的 PC
20 TaskIdNext:
21     LD  R1, TaskId
22     LDI R2, 1
23     ADD R1, R1, R2        ; TaskId+1
24     ST  R1, TaskId        ; TaskId = R3
25     LD  R2, TaskMax
26     CMP R1, R2            ; if (TaskId >= TaskMax)
27     JLT LoadTask
28     ST  R0, TaskId        ;     TaskId = 0
29 LoadTask:                 ; 載入新行程暫存器
30     CALL GetTaskBuf
31     LD  R9, [R1+36]
32     LDI R12, 192          ; 允許軟硬體中斷 (I:IRQ, T:
33     SWI 0
34     SWI 5                 ; 重設計時器，避免中斷太快發
35     LD  R15, [R1+60]      ; 載入新行程的 PC 以切換到新
36 GetTaskBuf:
37     ST  R2, TR2
38     LDI R1, 64            ; R1=64
39     LD  R2, TaskId        ; R2=TaskId
40     MUL R1, R2, R1        ; R1=TaskId*64
41     LD  R2, TaskBufPtr    ; R2=TaskBufPtr
42     ADD R1, R2, R1        ; R1=&TaskBuf[TaskId
43     LD  R2, TR2
44     RET
45 Task1:                    ; 第一個行程 Task1,
46     LD  R9, Task1Ptr      ; 顯示 Task1
47 FOR1:
48     SWI 3
49     JMP FOR1
50 Task2:                    ; 第二個行程 Task2,
51     LD  R9, Task2Ptr      ; 顯示 Task2
52 FOR2:
53     SWI 3
54     JMP FOR2
55 TR1:         RESW 1
56 TR2:         RESW 1
57 TR14:        RESW 1
58 TaskId:      WORD 0          ; 目前行程代號
59 TaskMax:     WORD 2          ; 所有行程的數目
60 TaskBuf:                     ; 行程結構儲存區，用
61 TaskBuf1:    RESW 15
62 Task1PC:     WORD Task1
63 TaskBuf2:    RESW 15
64 Task2PC:     WORD Task2
65 TaskBufPtr:  WORD TaskBuf
66 Task1Name:   BYTE "Task1"
67 Task1Ptr:    WORD Task1Name
68 Task2Name:   BYTE "Task2"
69 Task2Ptr:    WORD Task2Name
```

EOS0 – 組譯

```
Assembler:asmFile=eos0.asm1 objFile=eos0.obj1
===============Assemble=============
InterruptVector:
    JMP ResetHandler
    JMP Unexpected
...
=================PASS1================
   0 InterruptVector:                          ff (null)
   0 JMP ResetHandler                        J 26 (null)
   4 JMP Unexpected                          J 26 (null)
...
=============PASS2==============
   0 InterruptVector:                          ff
   0 JMP ResetHandler                        J 26 26000010
   4 JMP Unexpected                          J 26 26000008
...
  90 Task1:                                    ff
  90 LD  R9, Task1Ptr                       L  0 009F00B1
  94 FOR1:                                     ff
  94 SWI 3                                  J 2a 2A000003
  98 JMP FOR1                               J 26 26FFFFF8
...
  a8 TR1:   RESW 1                          D f0 00000000
  ac TR2:   RESW 1                          D f0 00000000
...
==========Save to ObjFile:eos0.obj1==========
260000102600000826000004260000102 6FFFFFC08C000C02A0000022600007 0C
000000000000090000000000000000000000000000000000000000000000000 0
```

EOS0 – 執行

- 指令： vm1 eos0.obj1 10
- 說明：
  - 用虛擬機 vm1 執行 eos0, 每 10 個指令切換一次
    。



開放電腦計畫 – 編譯器 CC1

C1 語言範例 – sum.c1

```
int main() {
    int s = sum(10);
    return s;
}

int sum(int n) {
    int i, s;
    for (i=1; i<=10; i++)
        s += i;
    return s;
}
```

C1 語言範例 - sum.c1 剖析結果

```
D:\ocs\ss1>cc1 test.c1 test.asm1
compile file:test.c1
B     GLOBAL    003E5C10 00000000
T     int       003E5C98 003E5C10
T     float     003E5CF8 003E5C10
T     char      003E5D58 003E5C10
======= parsing ========
+PROG
 +METHOD
  +ETYPE
    KEY:int
  -ETYPE
   ID:sum
M     sum       003E62C0 003E5C10
   KEY:(
  +PARAM_LIST
   +PARAM
    +ETYPE
     KEY:int
    -ETYPE
    +VAR
     ID:n
V     n         003E6500 003E62C0 int:*0:[0]
    -VAR
   -PARAM
  -PARAM_LIST
   KEY:)
  +BLOCK
B               003E6608 003E62C0
    KEY:{
   +BASE
    +STMT
     +DECL
```

...

```
              -EXP
             -EXP_LIST
              KEY:)
            -ATOM
           -PATH
          -TERM
         -EXP
        -VAR
       -VAR_LIST
      -DECL
     -STMT
      KEY:;
    -BASE
    +BASE
     +STMT
      KEY:return
     +EXP
      +TERM
       +PATH
        +ATOM
         ID:s
V     s         003E82B8 003E8010 int:*0:[0]
        -ATOM
       -PATH
      -TERM
     -EXP
    -STMT
     KEY:;
   -BASE
    KEY:}
  -BLOCK
 -METHOD
-PROG
```

C1 語言的 EBNF 語法

1. PROG = (STRUCT | METHOD | DECL ;)*
2. METHOD = ETYPE ** ID(PARAM_LIST?) BLOCK
3. STRUCT = struct ID { DECL_LIST ; }
4. BLOCK = { BASE* }
5. BASE = IF | FOR | WHILE |

開放電腦計畫 - 已完成事項

- AS1 組譯器
- VM1 虛擬機
- CC1 剖析器 + 語意分析
- EOS0 最小嵌入式作業系統 (兩行程切換)

開放電腦計畫 – 後續工作（程式部份）

- 軟體部份
  - 完成編譯器 CC1 的程式碼產生動作
  - 撰寫 EOS1 嵌入式作業系統（參考 Jserv 的 CuRT)
  - 撰寫 OS1 作業系統（參考 UNIX v6)
  - 撰寫 LD1 連結器（非必要）
  - 撰寫 CI1 解譯器（非必要）
- 硬體部份
  - 用 Verilog 設計 CPU1
  - 在 Altera 的板子上實測 CPU1
  - 將 Arduino 的 CPU 換爲 FPGA ，並燒入 CPU1

開放電腦計畫 – 後續工作（文件部份）

- 撰寫下列書籍
  - 編譯器 – 理論與實務
  - 作業系統 – 理論與實務
  - 計算機結構 – 理論與實務
- 修改下列書籍
  - 系統程式 – 理論與實務