金門大學 資訊工程系 101 學年度下學期 系統程式 期中考 出題者：陳鍾誠

考生姓名：　　學號：　　得分：

以下題目均按學號規則推算出指定的題數來做，學號為奇數者做奇數題，偶數者做偶數題，做錯題目則該題不計分，共 99 分，1 分奉送！

名詞解釋：每題 3 分，做指定的 4 題，共 12 分。

1. 系統程式：系統程式是與作業系統或電腦系統相關的軟體。這些程式通常會使用系統呼叫與底層的硬體或作業系統溝通。
2. 系統軟體：系統軟體是程式設計師所用到的，通常是與作業系統或電腦系統相關的軟體。像是編譯器、組譯器、連結器、虛擬機等。
3. gcc：GNU C compiler, 開源組織 GNU 所推出的 c 語言編譯器，包含一整套編譯連結工具。
4. objdump：GNU 的目的檔傾印工具，可以用來列出目的檔資訊並進行反組譯動作。
5. ar：GNU 的函式庫建置管理工具，可以將很多目的檔封裝成單一個函式庫檔。
6. make：make 是 GNU 的專案建置工具，可用來解譯 Makefile 檔案並進行大型專案的自動建置工作。
7. cpu0：CPU0 是課本中所定義的一個 32 位元簡易處理器，是為了幫助學習者理解電腦架構所設計的，包含 R0~R15 等 16 個可存取暫存器。
8. ORG 假指令：通知組譯器將將程式計數器設到參數指定位址的一個假指令。

簡答題：每題 5 分，做指定的 4 題，共 20 分。

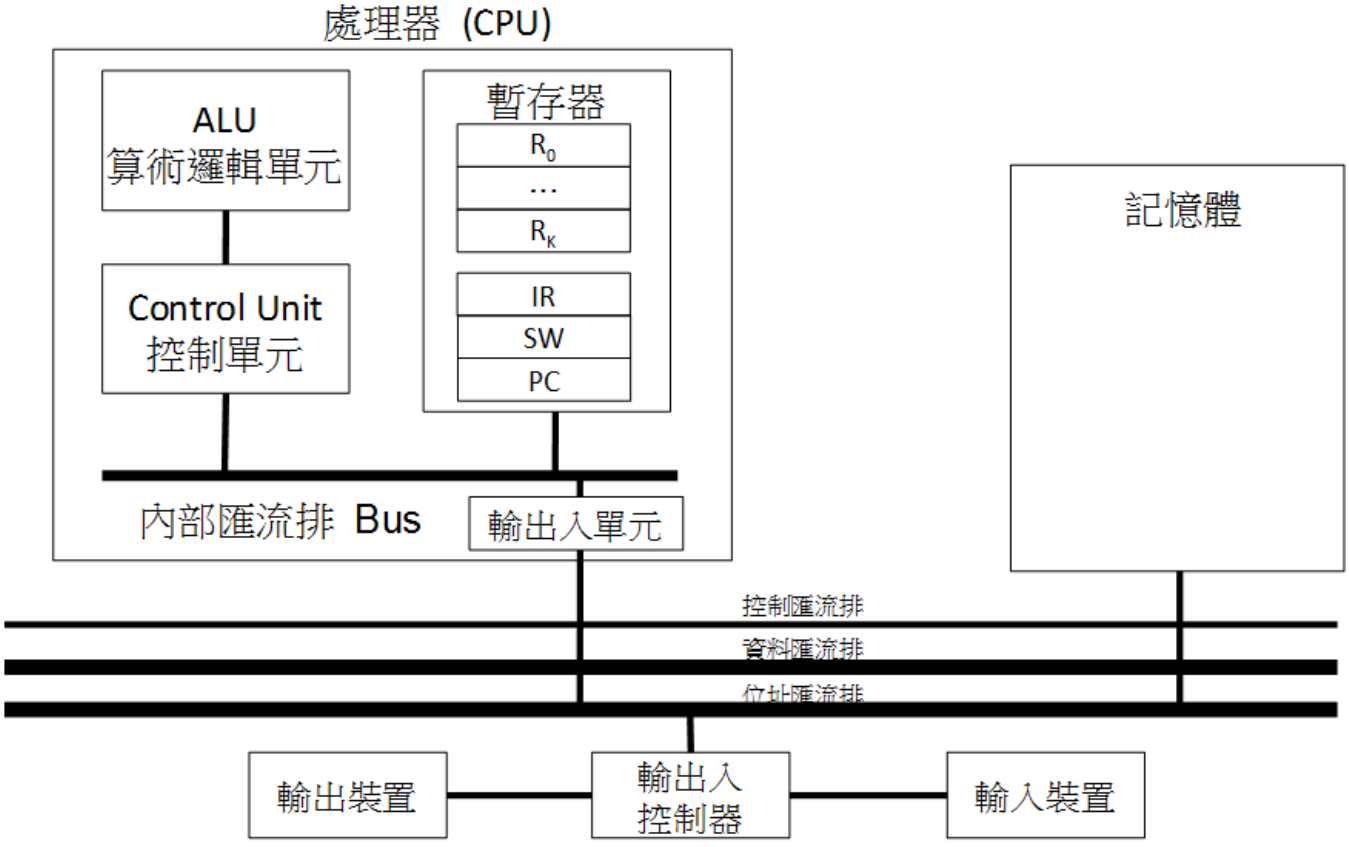
1. 請畫出電腦的馮紐曼架構，並說明其中各個元件的功用？

記憶體 Memory：儲存程式和資料，以供 CPU 讀取或寫入。

CPU：電腦指令運作的核心，可以解讀指令並執行對應的動作。

輸出入：包含輸出裝置 (螢幕、硬碟、...)、輸入裝置(鍵盤、滑鼠、硬碟...) 與輸出入控制器 (介面卡, ...)，讓 CPU 可以達成輸出入動作。

匯流排 BUS：連接 CPU, Memory, IO Device 的線路，可細分為「位址、資料與控制」等三種線路。



2. 請說明組譯器第一階段的功能？

第一階段：計算指令位址，並且記住所有符號的位址，建立符號表

3. 請說明組譯器第二階段的功能？

第二階段：將組合語言指令轉換成機器碼，並進行資料編碼的動作。

4. 請說明指令表在組譯器中的功能？

指令表：用來儲存指令的字串與代碼，讓組譯器可以查詢指令的代碼，並將指令字串轉為機器碼中的指令代碼。

5. 請說明符號表在組譯器中的功能？

符號表：用來儲存標記與變數的名稱與位址，讓組譯器可以查詢該標記的位址，以便將運算元中的標記轉換為機器碼。

6. 請說明連結器的功能為何？

連結器會將很多目的檔 (或函式庫) 連結在一起，形成一個完整的執行檔 (或函式庫) 後輸出。

7. 請說明動態連結器的功能為何？

動態連結器會在執行時期才進行連結動作，因此不需要事先將所有目的檔連結進來，比靜態連結器多了一些彈性。

8. 請說明載入器的功能為何？

載入器的功能是將目的檔 (機器碼) 載入到記憶體當中，然後將「程式計數器」(PC) 指向該程式的第一個指令位址，以便開始執行該程式。

程式題：每題 **10** 分，做指定的 **4** 題，共 **40** 分。

1. 請寫出一個組合語言程式，可以將變數 **Y** 的內容平方後放到變數 **Y2** 中。

```
     LD  R1, Y
     MUL R2, R1, R1
     ST  R2, Y2
     RET
Y:   WORD 5
Y2:  WORD 0
```

2. 請寫出一個組合語言程式，可以將變數 **X** 的內容加 **10** 後放回 **X** 中。

```
     LD  R1, X
     LDI R2, 10
     ADD R1, R1, R2
     ST  R1, X
     RET
X:   WORD 5
```

3. 請寫出一個組合語言程式，可以將變數 **X, Y** 的內容交換。

```
     LD  R1, X
     LD  R2, Y
     ST  R1, Y
     ST  R2, X
     RET
X:   WORD 5
Y:   WORD 3
```

4. 請寫出一個組合語言程式，可以取得變數 **X, Y** 中較大的值放入變數 **max** 中。

```
        LD  R1, X
        LD  R2, Y
        CMP R1, R2
        JGT SET
        ST  R2, max
        JMP EXIT
SET:    ST  R1, max
EXIT:   RET
X:      WORD 5
Y:      WORD 3
max:    WORD 0
```

5. 請寫出一個「組合語言副程式」，可以將 **R2, R3** 中較小者放入 **R1** 中。

```
        LD  R13, SPTR
        PUSH R14
        LDI R2, 3
        LDI R3, 5
        CALL MIN
        POP R14
        RET
STACK:  WORD 0
SPTR:   WORD STACK
// 以上為主程式不用寫，僅供參考！要寫的是以下部分
```

7. 請寫出一個組合語言程式，給定整數陣列 **A** 可以計算出 **A** 內容的總和放在 **SUM** 變數中。

```
SUM:    LDI   R1, 0      ; R1=0
        LD R2, aptr   ; R2=aptr
        LDI   R3, 3      ; R3=3
        LDI   R4, 4      ; R4=4
        LDI   R9, 1      ; R9=1
FOR:
        LD R5, [R2]   ; R5=*aptr
        ADD   R1,R1,R5   ; R1+=*aptr
        ADD   R2, R2, R4 ; R2+=4
        SUB   R3, R3, R9 ; R3--;
```

```
MIN:    CMP R2, R3
        JLT SET
        MOV R1, R2
        JMP EXIT
SET:    MOV R1, R3
EXIT:   RET
```

6. 請寫出一個「組合語言副程式」，可以設定 **R1** 為 3^{R2} 。

```
        LD  R13, SPTR
        PUSH R14
        LDI R2, 3
        CALL Power3
        POP R14
        RET
STACK:  WORD 0
SPTR:   WORD STACK

Power3: LDI R4, 1       ; R4 初始值為 1
        LDI R1, 1       ; R1 初始值為 1
        LDI R3, 3       ; R3 是常數 3
        LDI R5, 1       ; R5 是常數 1
LOOP:   CMP R4, R2      ; (R4 < R2) ?
        MUL R1, R1, R3  ; R1=R1*R3
        ADD R4, R4, R5  ; R4=R4+1
        JLT LOOP        ; goto LOOP
EXIT:   RET
```

```
        CMP R3, R0      ; if (R3!=0)
        JNE  FOR        ;   goto FOR;
        ST R1, sum      ; sum=R1
        LD R8, sum      ; R8=sum
        RET
a:      WORD  3, 7, 4   ; int a[]={3,7,4}
aptr:   WORD  a         ; int *aptr = &a
sum:    WORD  0         ; int sum = 0
```

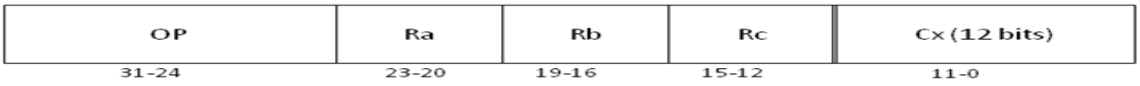
8. 請寫出一個組合語言程式，給定字串 **A**，可以將該字串複製到字串 **B** 中。

```
COPY:   LD R1, aptr
        LD  R2, bptr
        LDI R9, 1
LOOP:   LD  R3, [R1]
        ST  R4, [R2]
        CMP R3, 0
        ADD R1, R1, R9
        ADD R2, R2, R9
        JNE LOOP
        RET
A:      BYTE  72, 101, 108, 108, 111, 33, 0
; 也可以寫成 "Hello!", 0
B:      RESB    7
aptr:   WORD A
bptr:   WORD B
```

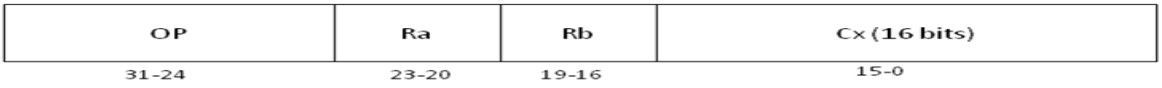
請寫出下列程式的位址欄與機器碼 (每列 **3** 分，位址欄 **1** 分，機器碼 **2** 分，做指定的 **9** 列，共 **27** 分)

題號	位址欄	組合語言	機器碼
1.	0000	LDI R1, 1	08100001
2.	0004	LD R2, aptr	002F003C
3.	0008	LD R3, bptr	003F003C
4.	000C	LD R4, len	004F0024
5.	0010	ADD R3, R3, R4	13334000
6.	0014	FOR: CMP R3, R0	10304000
7.	0018	JLT EXIT	22000014
8.	001C	LD R5, [R1]	00510000
9.	0020	ST R5, [R2]	01520000
10.	0024	ADD R2, R2, R1	13221000
11.	0028	SUB R3, R3, R1	14331000
12.	002C	JMP FOR	26FFFFE4
13.	0030	EXIT: RET	2C000000
14.	0034	len: WORD 6	00000006
15.	0038	a: BYTE 3, 2, 7, 6, 8, 5	030207060805
16.	003E	b: RESB 6	000000000000
17.	0044	aptr: WORD a	00000038
18.	0048	bptr: WORD b	0000003E

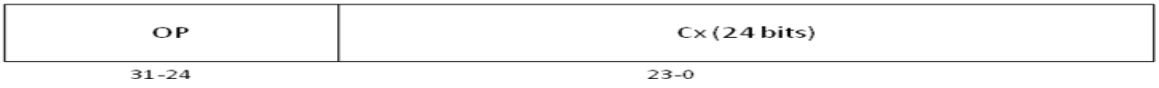
A 型



L 型



J 型



類型	格式	指令	OP	說明	語法	語意
載入儲存	L	LD	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 的暫存器版	LDR Ra, [Rb+Rc]	Ra→(byte)[Rb+ Rc]
	A	STR	05	ST 的暫存器版	STR Ra, [Rb+Rc]	Ra→[Rb+ Rc]
	A	LBR	06	LDB 的暫存器版	LBR Ra, [Rb+Rc]	Ra←(byte)[Rb+ Rc]
	A	SBR	07	STB 的暫存器版	SBR Ra, [Rb+Rc]	Ra→(byte)[Rb+ Rc]
	L	LDI	08	立即載入	LDI Ra, Rb+Cx	Ra← Rb + Cx
運算指令	A	CMP	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	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
跳躍指令	J	JEQ	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
	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	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
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)