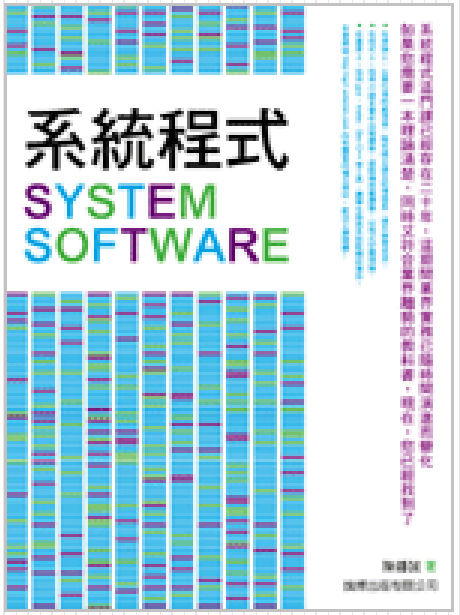
第 9 章、虛擬機器

作者：陳鍾誠

旗標出版社



第 9 章、虛擬機器

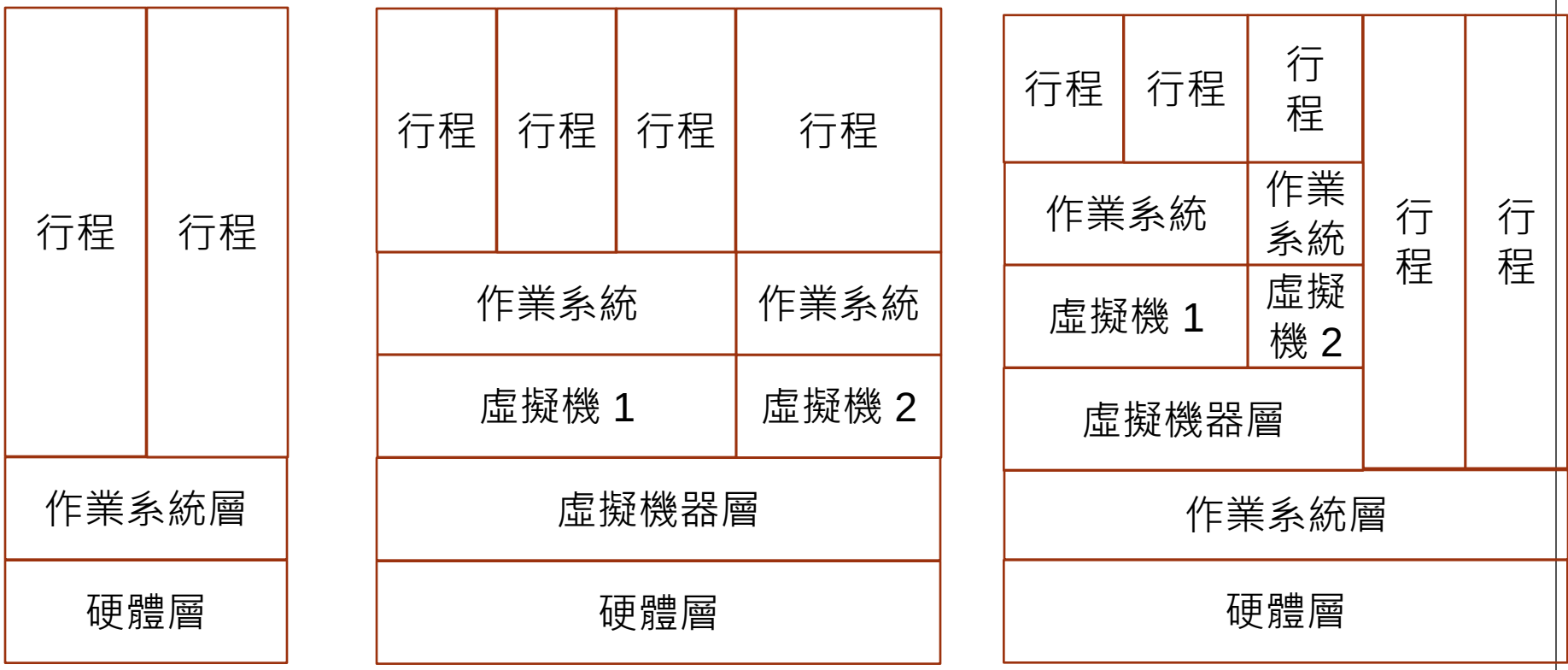
- 9.1　簡介
- 9.2　中間碼
- 9.3　CPU0 的虛擬機
- 9.4　實務案例 (一)：Java 的 JVM 虛擬機
- 9.5　實務案例 (二)：微軟的 Virtual PC 虛擬機

9.1 簡介

- 狹義的虛擬機 (Virtual Machine)
  - 虛擬機：模擬處理器指令集的軟體
  - 模擬器：模擬電腦行為的軟體

- 廣義的虛擬機
  - 然而, 有些軟體既會模擬指令集, 又會模擬電腦的行為, 像是 VMWare、Virtual PC、Virtual Box 這樣的軟體, 也被我們視為是一種虛擬機器。
  - 在大部分的情況下, 在我們不需要去區分虛擬機與模擬器的時候, 我們會使用虛擬機器一詞統稱所有的模擬程式, 而不去使用模擬器一詞。

圖 9.1原生與寄生式的虛擬機



(a) 無虛擬機時　(b) 原生式虛擬機　(c) 寄生式虛擬機

虛擬機器的範例

- 原生式虛擬機 (Native VM)
  - IBM 的 CP-40
  - IBM 的 System/370
- 寄生式虛擬機 (Hosted VM)
  - VMWare、Virtual PC、Virtual Box、Wine、Bochs、Qemu
- 程序虛擬機 (Process Virtual Machine)
  - 又稱為應用層虛擬機 (Application Virtual Machine)
  - 一種虛擬碼的解譯器
    - Java 的 Java Virtual Machine (JVM)
    - 微軟 .NET 所使用的 CLR 虛擬機器

9.2 中間碼

- 功能
  - 如果我們設計出與機器架構無關的中間碼
  - 然後利用虛擬機執行該中間碼 (p-code)
  - 此種中間碼將具有跨平台能力

- 範例
  - JAVA 的 bytecode ：可在 JVM 虛擬機中執行
  - 微軟的 MSIL：可在 .NET 平台中執行

三種虛擬機架構

- 記憶體機 (Memory Machine)
  - 可以直接對記憶體變數進行運算
- 暫存器機 (Register Machine)
  - 必須將變數載入暫存器中，才能進行運算
- 堆疊機 (Stack Machine)
  - 取出堆疊上層元素進行運算
  - 結果存回堆疊之中

範例 9.1 三種虛擬機的組合語言

▶範例 9.1 三種虛擬機的組合語言

(a) 記憶體機

```
    =    0     sum
    =    0     i
FOR0:
    CMP  i     10
    J    >     _FOR0
    +    sum   i    T0
    =    T0    sum
    +    i     1    i
    J          FOR0
_FOR0:
    RET        sum
```

(b) 暫存器機

```
        LDI     R1, 0
        ST      R1, sum
        LDI     R2, 0
        ST      R2, i
        LDI     R3, 1
        LDI     R4, 10
FOR0:
        CMP     R2, R4
        JGT     _FOR0
        ADD     R1, R1, R2
        ADD     R2, R2, R3
        JMP     FOR0
FOR0:
        ST      R1, sum
        RET
i:      RESW    1
sum:    RESW    1
```

(c) 堆疊機

```
        PUSH 0
        POP sum
        PUSH 0
        POP i
FOR0:
        PUSH i
        PUSH 10
        CMP
        PUSH _FOR0
        JGT
        PUSH sum
        PUSH i
        ADD
        POP sum
        PUSH i
        PUSH 1
        ADD
        POP i
        PUSH FOR0
        JMP
_FOR0:
        PUSH sum
        RET
i:      RESW    1
sum:    RESW    1
```

堆疊機

- 所有參數都存入堆疊當中, 然後才進行運算

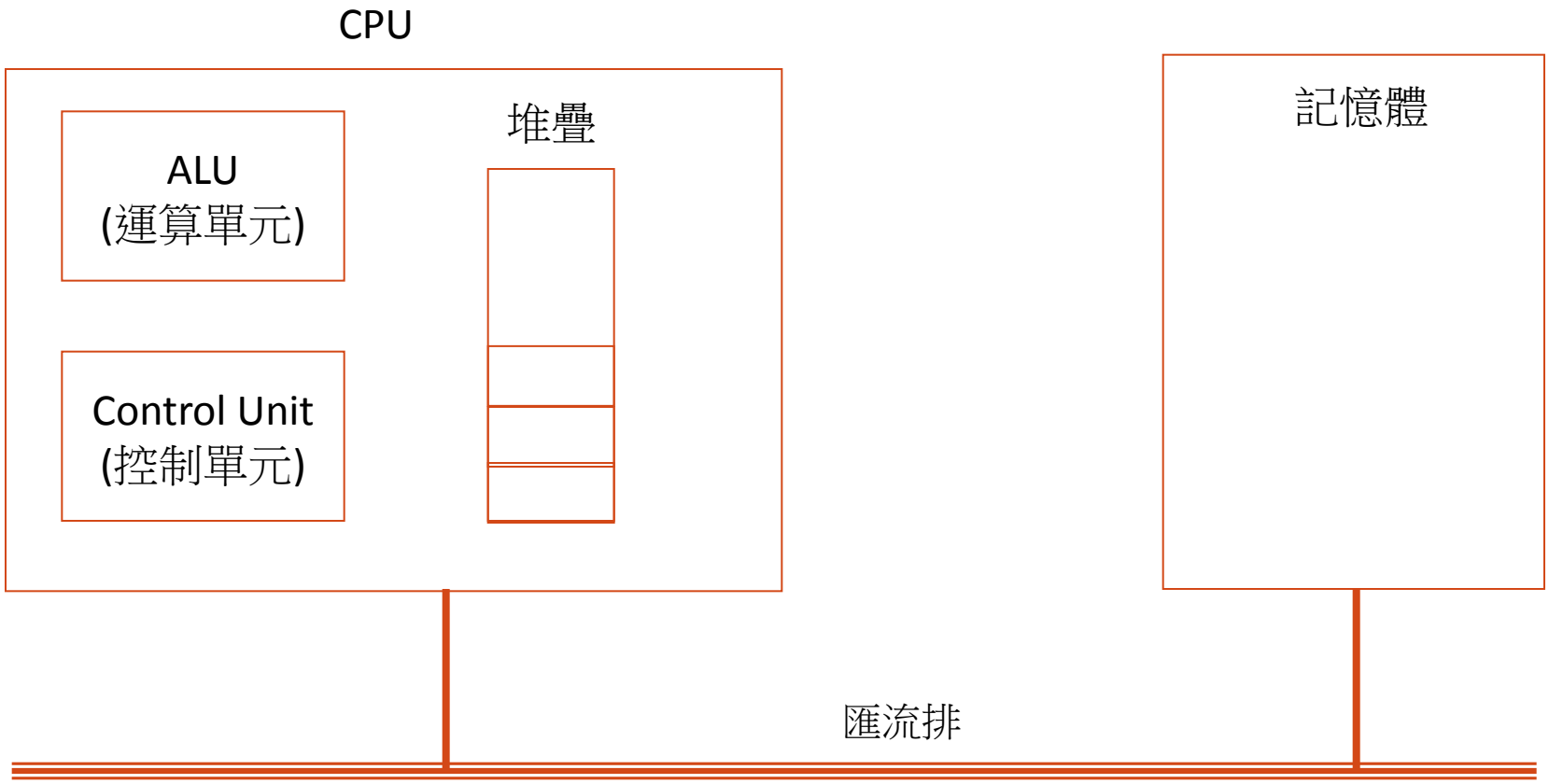


圖 9.2堆疊機 (Stack Machine) 的架構圖

範例 9.2 堆疊機的組合語言

▶範例 9.2 堆疊機的組合語言

堆疊機組合語言	說明
PUSH 6	推入 6 到堆疊中
PUSH 2	推入 2 到堆疊中
PUSH 3	推入 3 到堆疊中
ADD	將 3 與 2 相加 = 5
SUB	將 6 與 5 相減 = 1

圖 9.3 堆疊機執行過程

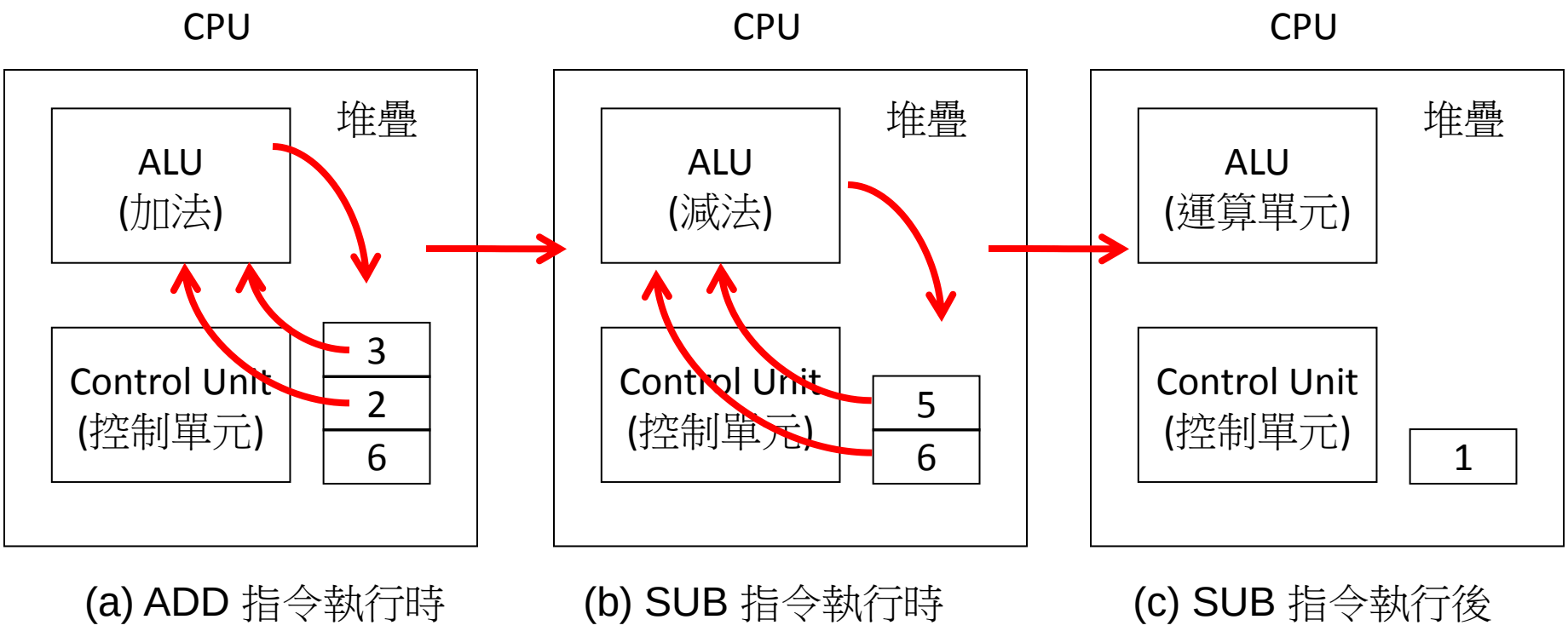


圖 9.3<範例 9.2> 的堆疊機執行過程

9.3 CPU0 的虛擬機

- 功能
  - 以軟體的方式模擬 CPU0 的執行過程
- 實作
  - 解譯 CPU0 的機器指令
  - 根據指令的意義，模擬對應的動作。

圖 9.4 CPU0 的虛擬機之演算法 (資料結構)

- 資料結構
  - 暫存器
    - IR, R[0~15]

▶圖 9.4 CPU0 的虛擬機之演算法

CPU0 虛擬機器的演算法	說明
// Virtual Machine for CPU0	CPU0 的模擬器類別
// global variable	共用變數
byte m[]	m 為記憶體
int32 IR	IR 為指令暫存器
int32 R[16]	R[] 為一般暫存器
PC is R[15]	PC 為 R[15] 的別名
LR is R[14]	LR 為 R[14] 的別名
SP is R[13]	SP 為 R[13] 的別名

圖 9.4 CPU0 的虛擬機之演算法 (指令提取)

- 指令擷取階段，將指令從記憶體取到 IR

```
// function
 Algorithm run(objFile)
   startAddress = readObjFile(objFile, m)
   PC = startAddress
   LR = -1
   stop = false
   while not stop
    R[0] = 0
    load IR from m[PC..PC +3]
    PC=PC+4
```

函數區
模擬執行目的檔
讀取目的檔到記憶體中
設定程式計數器為起始位址
設定連結暫存器為-1, 跳離用
設定停止旗標為尚未停止
進入迴圈, 開始執行指令
R[0] 暫存器永遠為 0
擷取指令到 IR 暫存器中
將 PC 加 4 (下一個指令)

圖 9.4 CPU0 的虛擬機之演算法 (指令解碼)

```
op = bits(IR, 24..31)                          取出指令碼 op
ra = bits(IR, 20..23)                          取出暫存器代號 ra
rb = bits(IR, 16..19)                          取出暫存器代號 rb
rc = bits(IR, 11..15)                          取出暫存器代號 rc
c5 = bits(IR, 0..4);                           取出位元 0..5 放入 c5 中
c12 = bits(IR, 0, 11)                          取出位元 0..11 放入 c12 中
c16 = bits(IR, 0, 15)                          取出位元 0..15 放入 c16 中
c24 = bits(IR, 0, 23)                          取出位元 0..23 放入 c24 中
if (bits(IR, 11, 11)!=0) c12 |= 0xFFFFF000;    處理 c12 可能為負數的情況
if (bits(IR, 15, 15)!=0) c16 |= 0xFFFF0000;    處理 c16 可能為負數的情況
if (bits(IR, 23, 23)!=0) c24 |= 0xFF000000;    處理 c24 可能為負數的情況
caddr = R[rb]+c16                              計算 LD, ST, … 的位址欄
raddr = R[rb]+R[rc]                            計算 LDR, STR, … 的位址欄
```

圖 9.4 CPU0 的虛擬機之演算法 (載入儲存指令)

```
switch op                                          根據指令碼 op 跳到對應程式
  LD: load R[ra] from m(caddr..caddr+3)            處理 LD 指令
  ST: store R[ra] into m(caddr..caddr+3)           處理 ST 指令
  LDB : load R[ra] form m[caddr]                     處理 LDB 指令
  STB : store R[ra] into m[caddr]                    處理 STB 指令
  LDR : load R[ra] from m[raddr..raddr+3]            處理 LDR 指令
  STR : store R[ra] into m[raddr..raddr+3]           處理 STR 指令
  LBR : load R[ra] from m[raddr]                     處理 LBR 指令
  SBR : store R[ra] into m[raddr]                    處理 SBR 指令
  LDI : R[ra] = cx                                   處理 LDI 指令
```

圖 9.4 CPU0 的虛擬機之演算法 (運算指令)

```
CMP : cc = compare(R[ra], R[rb]);          處理 CMP 指令
      set cc into bits(SW, 31..30)
MOV : R[ra] = R[rb]                        處理 MOV 指令
ADD : R[ra] = R[rb]+R[rc]                  處理 ADD 指令
SUB : R[ra] = R[rb]-R[rc]                  處理 SUB 指令
MUL : R[ra] = R[rb]*R[rc]                  處理 MUL 指令
DIV : R[ra] = R[rb] / R[rc]                處理 DIV 指令
AND : R[ra] = R[rb] and R[rc]              處理 AND 指令
OR  : R[ra] = R[rb] or R[rc]               處理 OR 指令
XOR : R[ra] = R[rb] xor R[rc]              處理 XOR 指令
ROL : R[ra] = R[rb] rol c5                 處理 ROL 指令
ROR : R[ra] = R[rb] ror c5                 處理 ROR 指令
SHL : R[ra] = R[rb] shl c5                 處理 SHL 指令
SHR : R[ra] = R[rb] shr c5                 處理 SHR 指令
```

圖 9.4 CPU0 的虛擬機之演算法 (跳躍指令)

```
JEQ : if (cc is =) PC = PC +c24
JNE : if (cc is not = ) PC = PC+ c24
JLT : if (cc is <) PC = PC+ c24
JGT : if (cc is >) PC = PC+ c24
JLE : if (cc in {<, =}) PC = PC+c24
JGE : if (cc in {>, =}) PC = PC+ c24
JMP : PC = PC+c24
SWI : R[14] = PC; PC= c24
JSUB : R[14] = PC; PC=PC+ c24
RET :
  if (R[14]<0)
    stop=true
  else
    PC=LR
```

```
處理 JEQ 指令
處理 JNE 指令
處理 JLT 指令
處理 JGT 指令
處理 JLE 指令
處理 JGE 指令
處理 JMP 指令
處理 SWI 指令
處理 JSUB 指令
處理 RET 指令
如果連結暫存器<0
  則跳離
否則
  返回上一層
```

圖 9.4 CPU0 的虛擬機之演算法 (堆疊指令)

```
      PUSH : SP= SP-4;store R[ra] into m[SP..SP+3]
      POP : SP=SP+4;load R[ra] from m[SP..SP+3]
      PUSHB : SP=SP-1;store R[ra] into m[SP]
      POPB : SP=SP+1;load R[ra] from m[SP]
    end switch
    print PC, IR, R[ra]
  end while
  dumpRegisters()
End Algorithm
```

處理 PUSH 指令
處理 POP 指令
處理 PUSHB 指令
處理 POPB 指令

印出相關暫存器以便觀察

印出所有暫存器以便觀察

9.4 實務案例 (一)：Java 的 JVM 虛擬機

- JVM (Java Virtual Machine)
  - JVM 是 Java 的虛擬機
  - 開放原始碼界還有
    - Kaffe、Jikes、Mono, Apache Harmony, Google Dalvik, …

範例 9.3 Java 的程式與編譯執行過程

範例 9.3 Java 的程式與編譯執行過程

(a) Java 程式 HelloWorld.java

```
class HelloWorld {
   public static void main (String[] args) {
      System.out.println("Hello World!");
   }
}
```

(b) 編譯與執行過程

```
D:\ch10>javac HelloWorld.java

D:\ch10>java HelloWorld
Hello World!
```

圖 9.5 Java 程式的編譯與執行方式

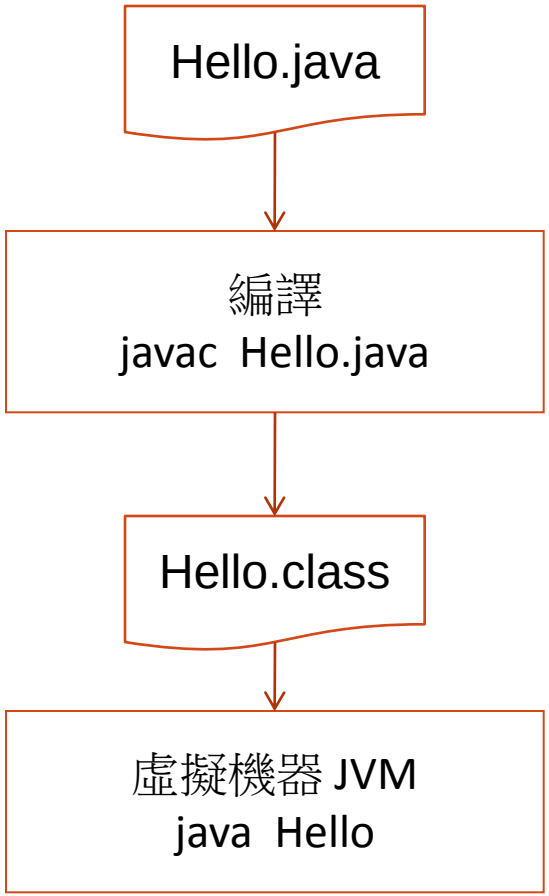
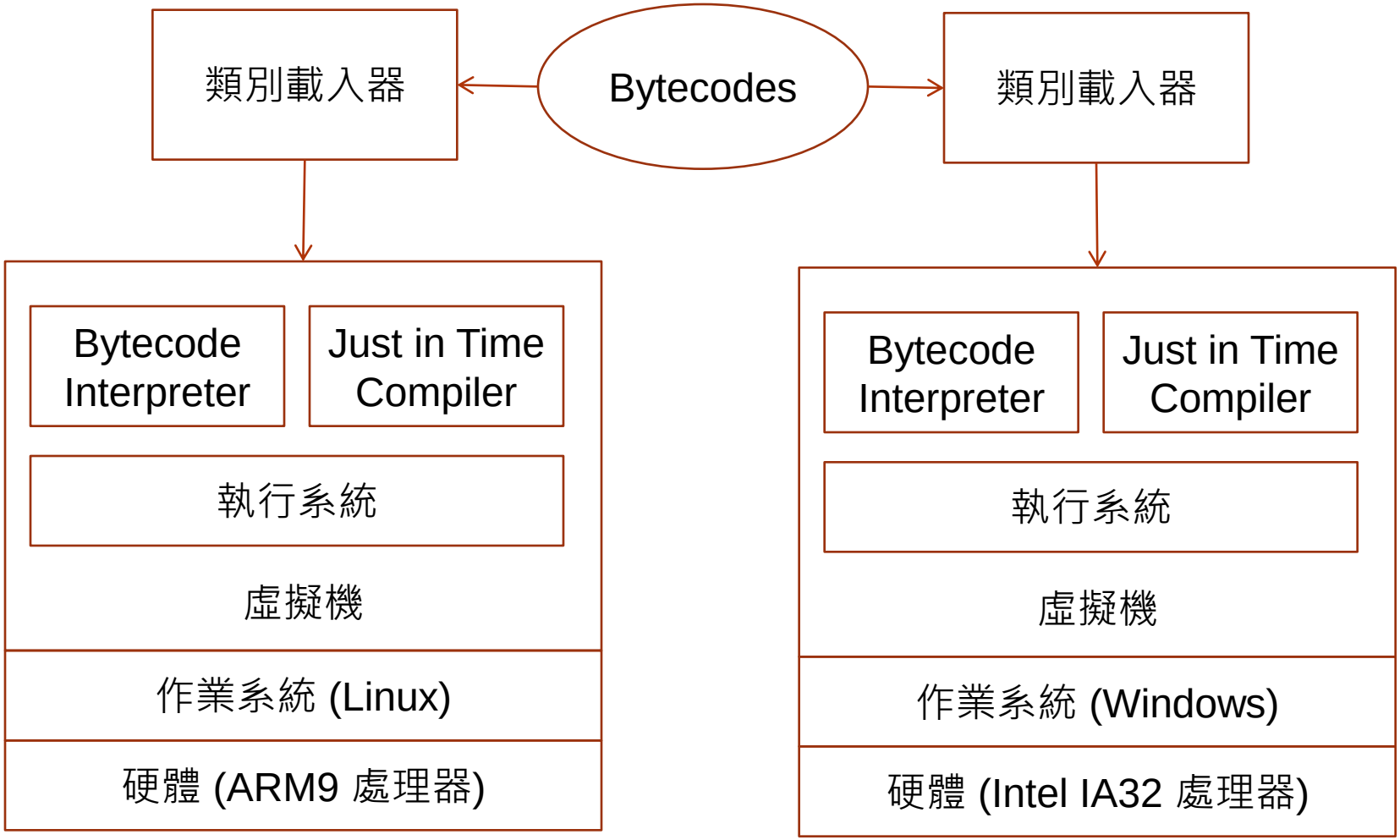


圖 9.6 利用虛擬機讓 Bytecode 跨平台執行



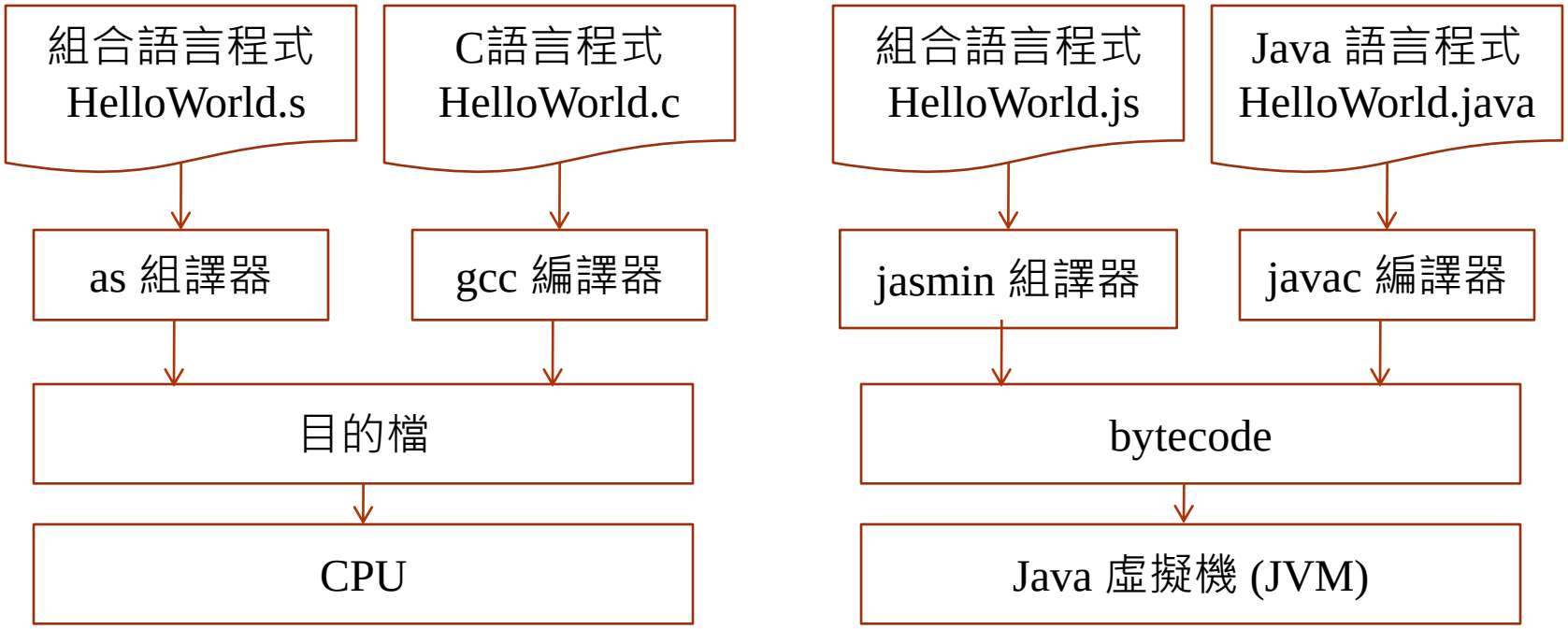
Java 當中動態載入技術的範例

- 範例 9.4
  - 動態的載入一個由變數 name 所指定的 byte code 檔案, 然後利用該檔案建立出對應的物件。
  - 這樣的技術讓 Java 程式可以在詢問使用者之後, 再決定要載入哪一個類別
  - 不需要在一開始時就載入所有的函式庫, 達成動態連結與動態載入的效果。

範例 9.4 Java 當中動態載入技術的範例

```
...
Class type=ClassLoader.getSystemClassLoader().loadClass(name);
Object obj = type.newInstance();
...
```

圖 9.7 Java 與 C 程式的執行模式對照



(a) C 程式的執行方式

(b) Java 程式的執行方式

利用 javap 指令將 bytecode 反組譯成組合語言

範例 9.5 利用 javap 指令將 bytecode 反組譯成組合語言

```
> javap -c HelloWorld
Compiled from "HelloWorld.java"
class HelloWorld extends java.lang.Object{
HelloWorld();
 Code:
  0:   aload_0
  1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
  4:   return

public static void main(java.lang.String[]);
 Code:
 0:   getstatic  #2; //Field java/lang/System.out:Ljava/io/
PrintStream;
 3:   ldc      #3; //String Hello World!
 5:   invokevirtual #4;//Method java/io/PrintStream.
println:(Ljava/lang/String;)V
 8:   return
}
```

9.5 實務案例 (二)：微軟的 Virtual PC 虛擬機

- X86 上常用的虛擬機
  - VMWare
  - Virtual PC
  - Virtual Box
  - Bochs
  - QEMU
- 本節使用 Virtual PC 作為範例

圖 9.8在 Microsoft Windows 當中以 Virtual PC 軟體執行 Red Hat 9.0 的情況

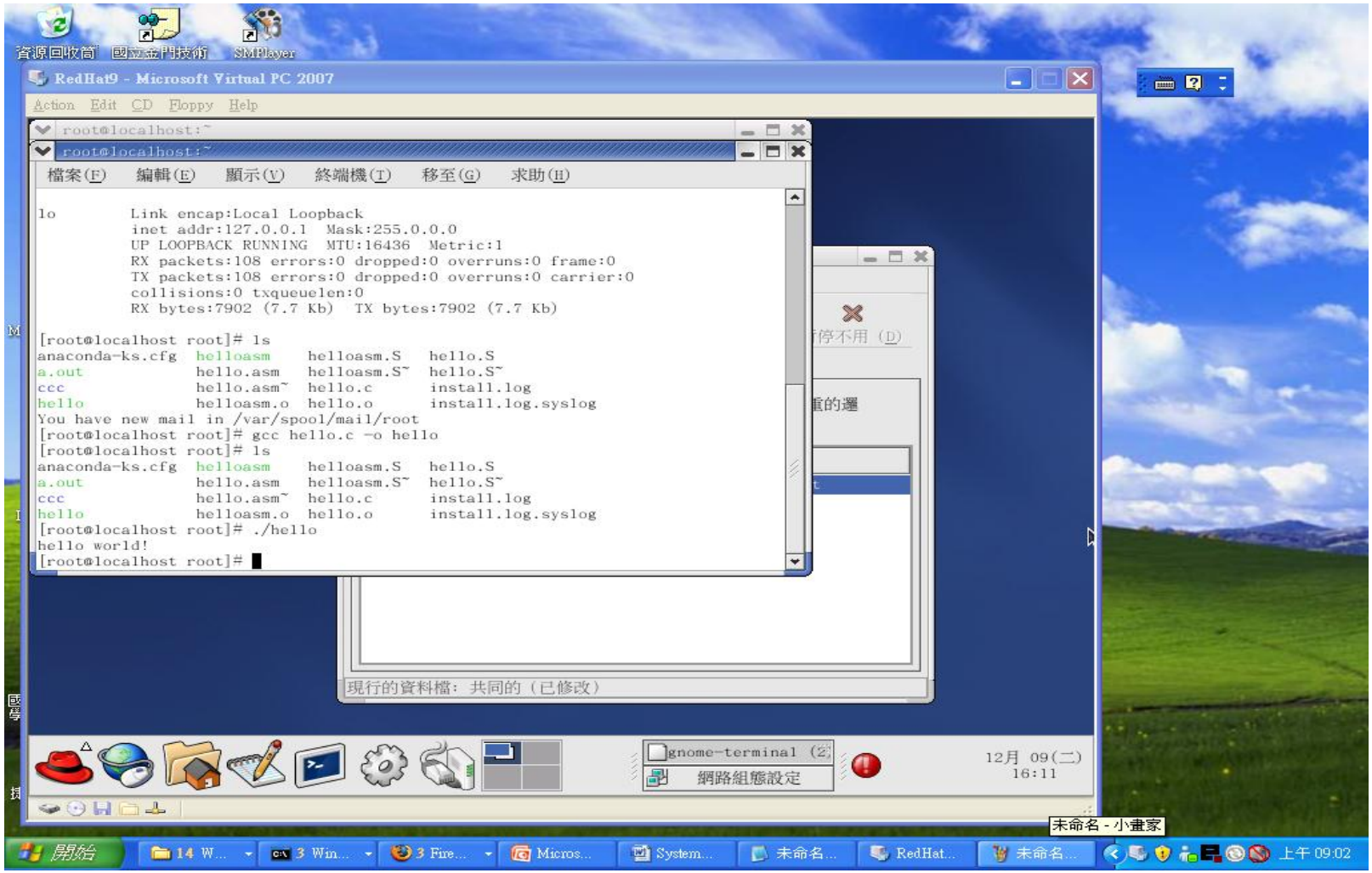


圖 9.9 Virtual PC 的啟動視窗

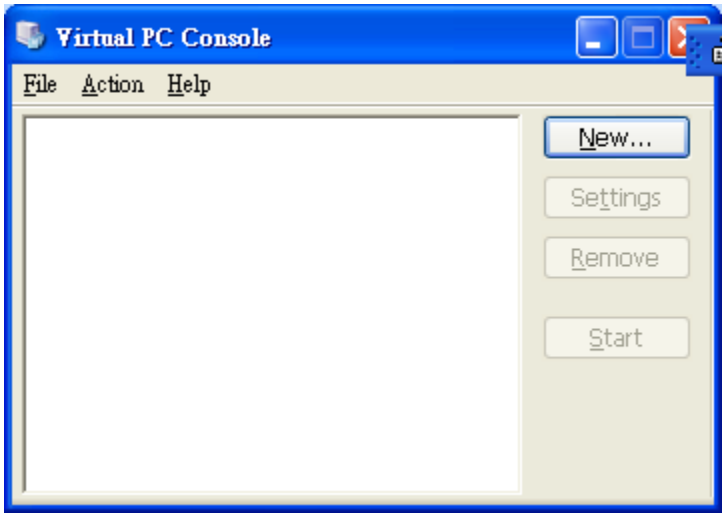


圖 9.10 Virtual PC 虛擬機的存檔畫面

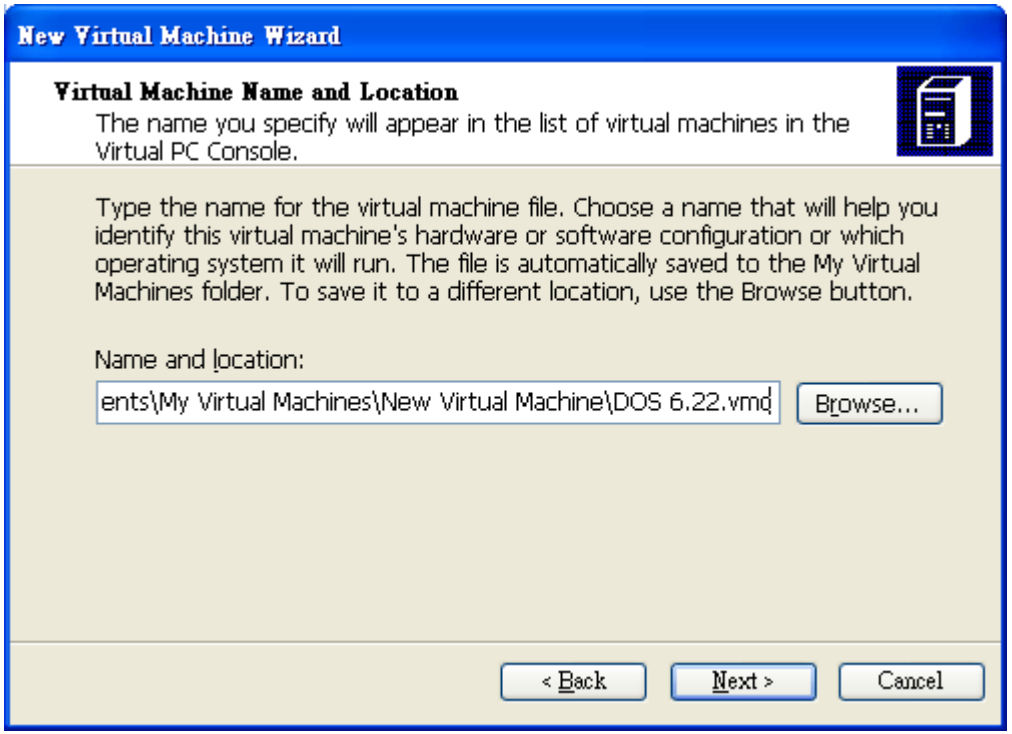


圖 9.11 選擇 A new virtual hard disk 以建立新的虛擬硬碟。

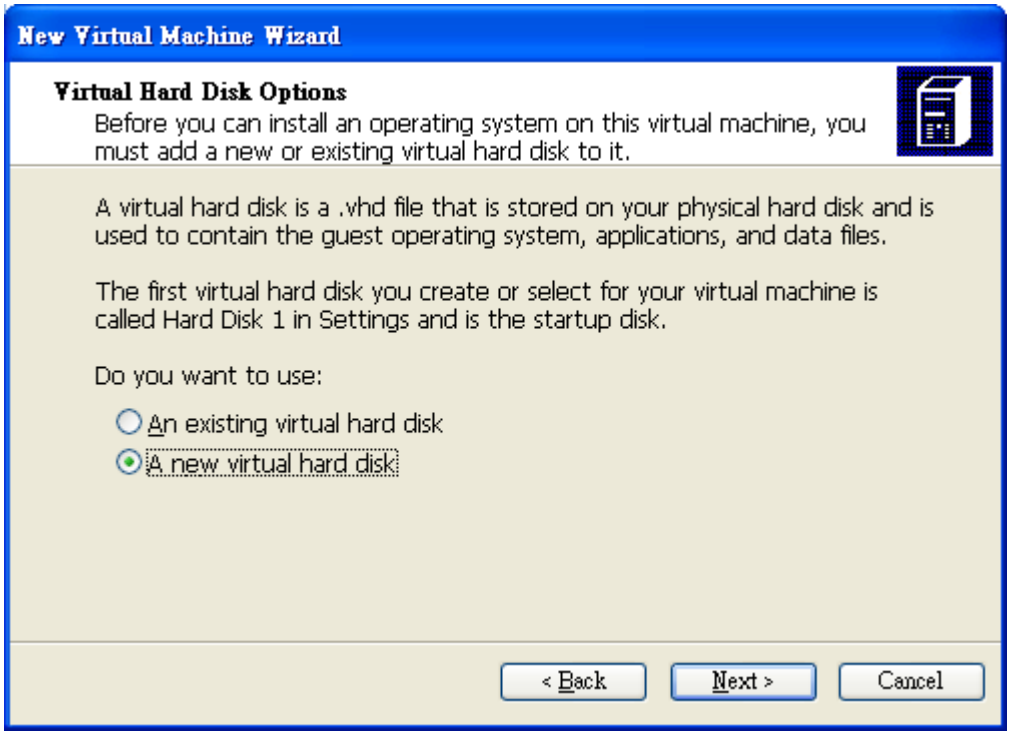


圖 9.12 請按下 Start 鍵啟動 DOS 虛擬機

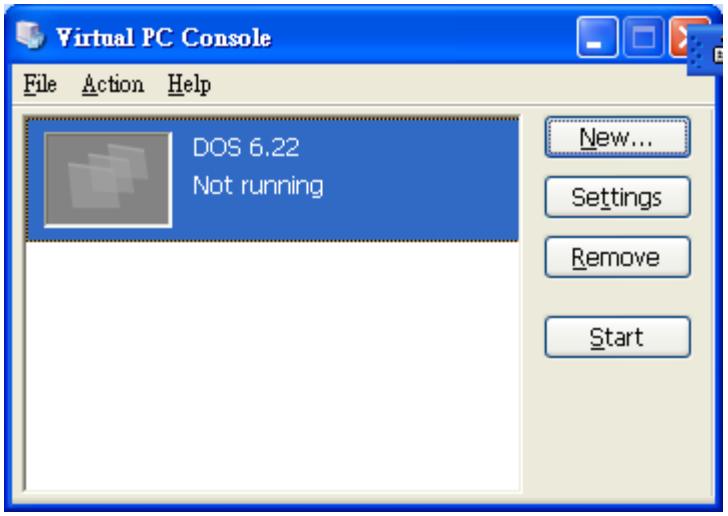


圖 9.13 Virtual PC 的虛擬機之啟動畫面

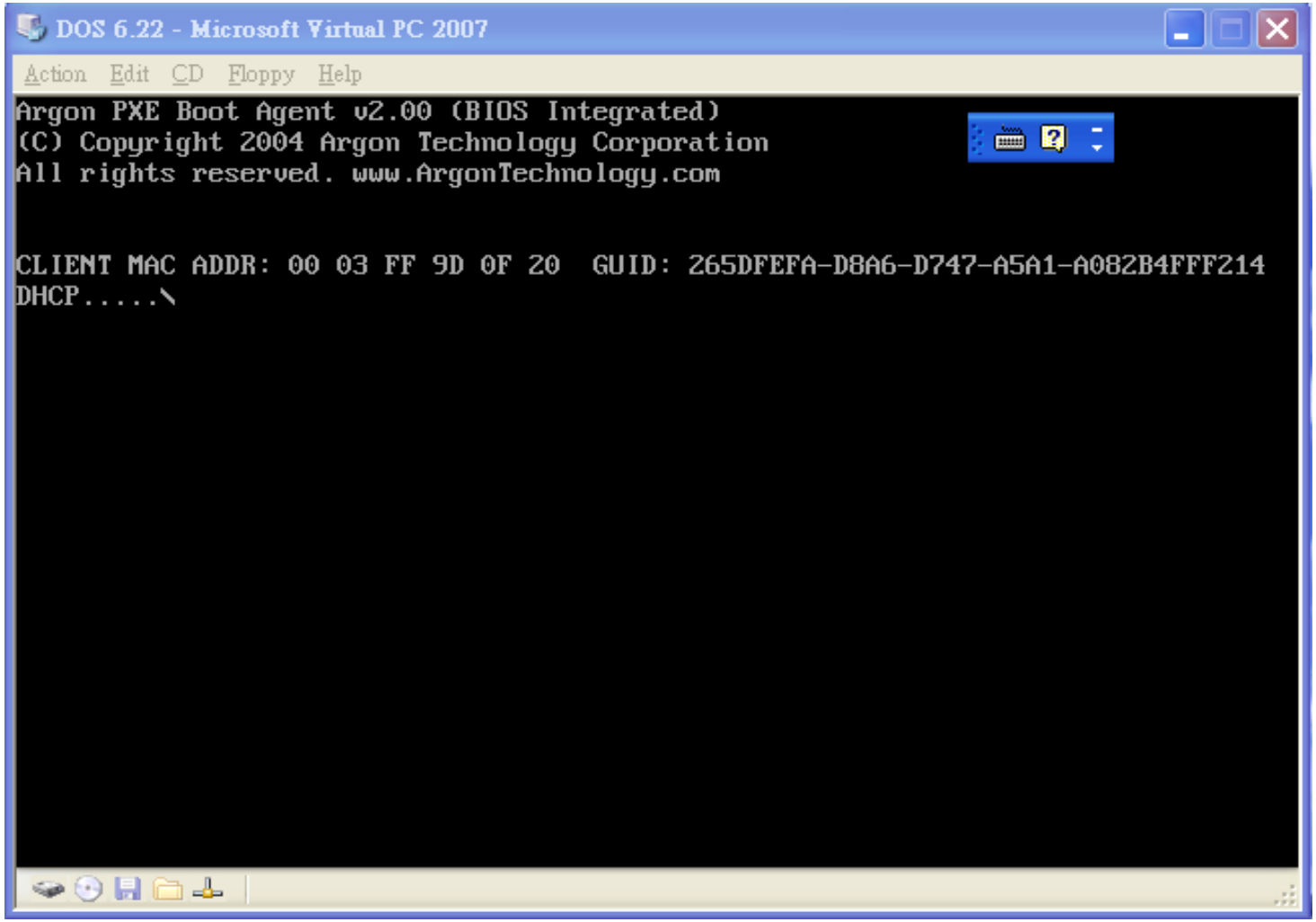


圖 9.14 在 Virtual PC 中指定 DOS 的軟碟開機映像檔

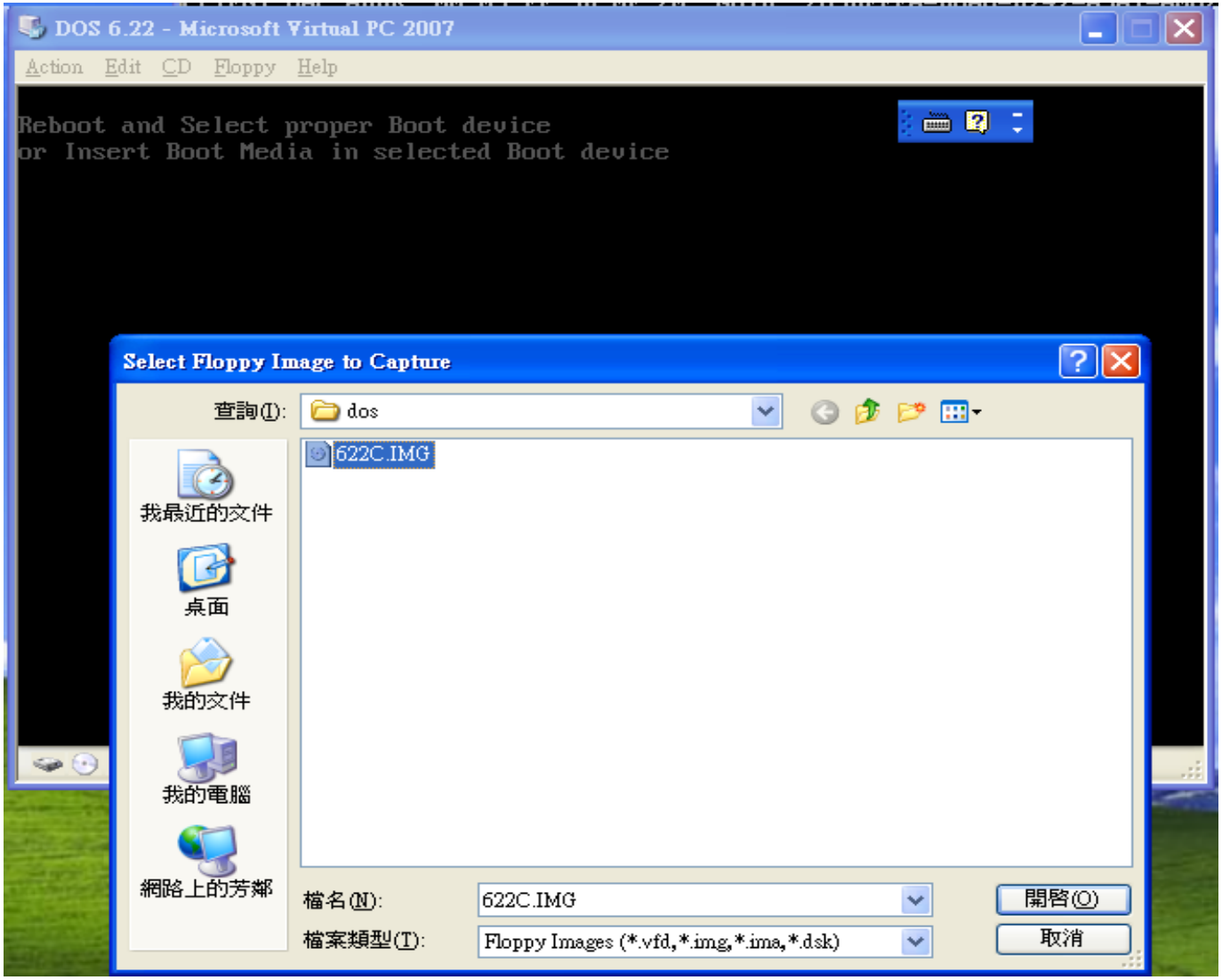
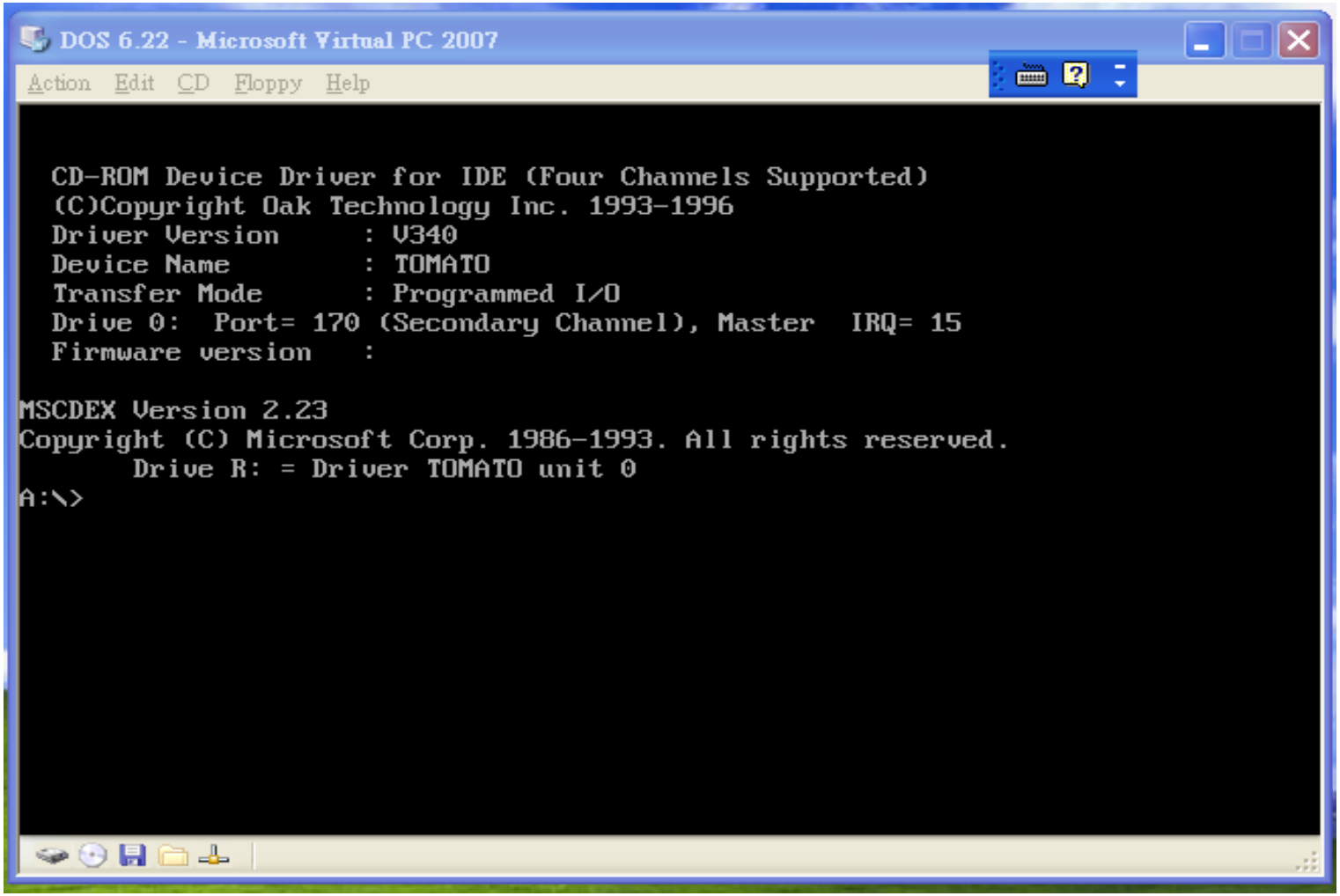


圖 9.15 在 Virtual PC 2007 中的 DOS 虛擬機開機畫面



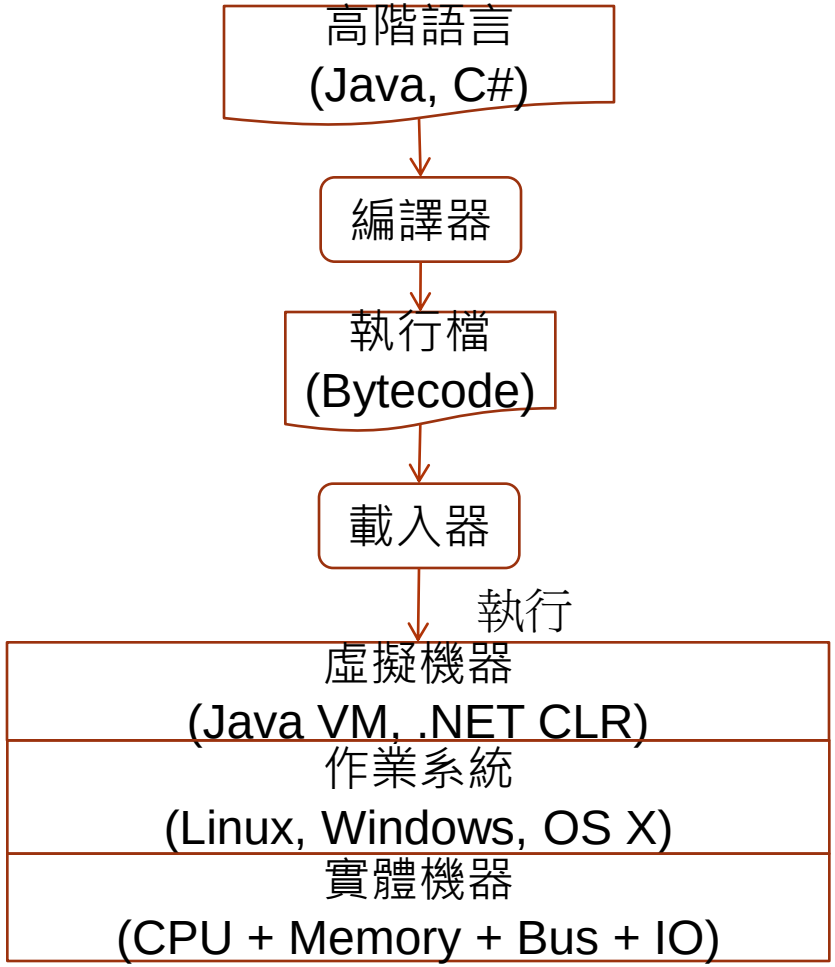
結語

- 虛擬機器
  - 虛擬機：模擬處理器指令集的軟體
  - 模擬器：模擬電腦行為的軟體
- 虛擬機的類型
  - 原生式 v.s. 寄生式
  - 1. 記憶體機 2. 暫存器機 3. 堆疊機
- CPU0 虛擬機的實作
- Java 的虛擬機
- Virtual PC 虛擬機器

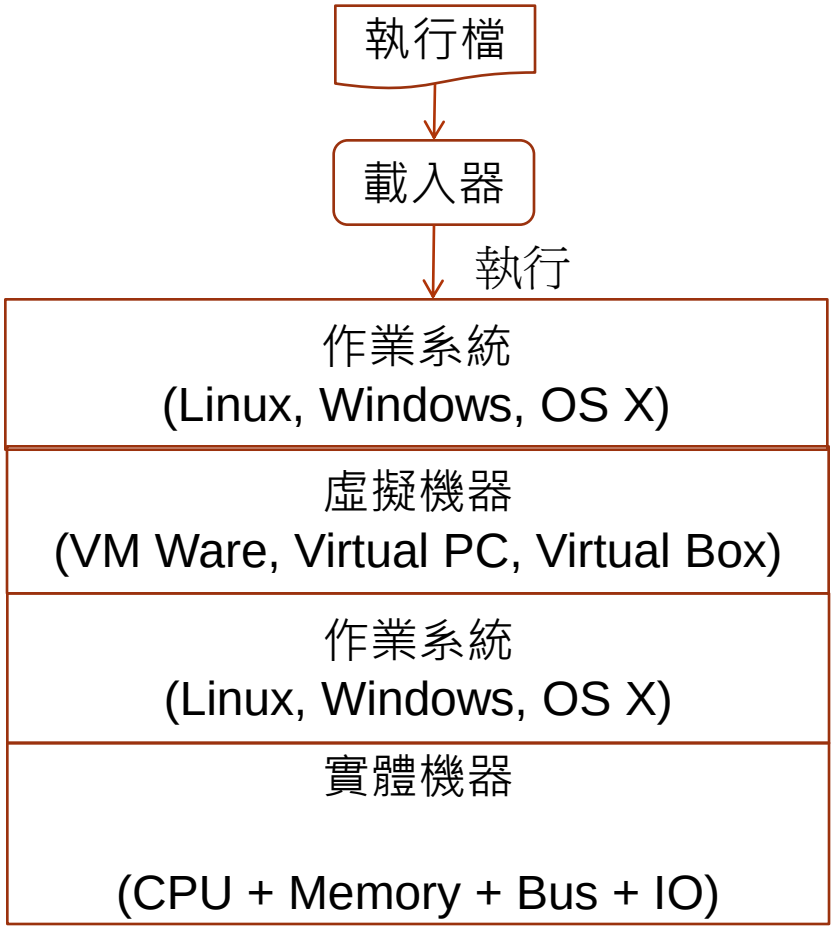
習題

- 9.1 請說明何謂虛擬機器？
- 9.2 請說明何謂記憶體機？
- 9.3 請說明何謂堆疊機？
- 9.4 請說明何謂暫存器機？
- 9.5 請閱讀本書的第 12 章, 並取得 ch12/CPU0.c 這個程式, 看看這個虛擬機是如何設計的。
- 9.6 請寫出一個 Java 程式, 並且使用 javac 編譯該程式, 然後使用 java 指令執行該程式。
- 9.7 接續上一題, 請使用 javap 將上一題產生的 bytecode 反組譯, 並
- 分析反組譯後的程式碼？
- 9.8 請安裝 Virtual PC, 然後在其中安裝 DOS 作業系統。

未包含於書中的圖片



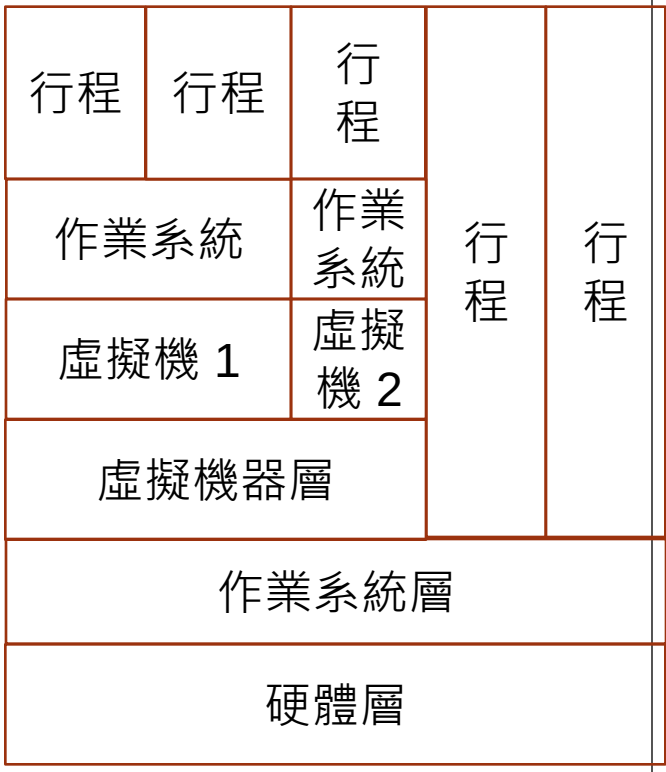
(a) 程序式虛擬機



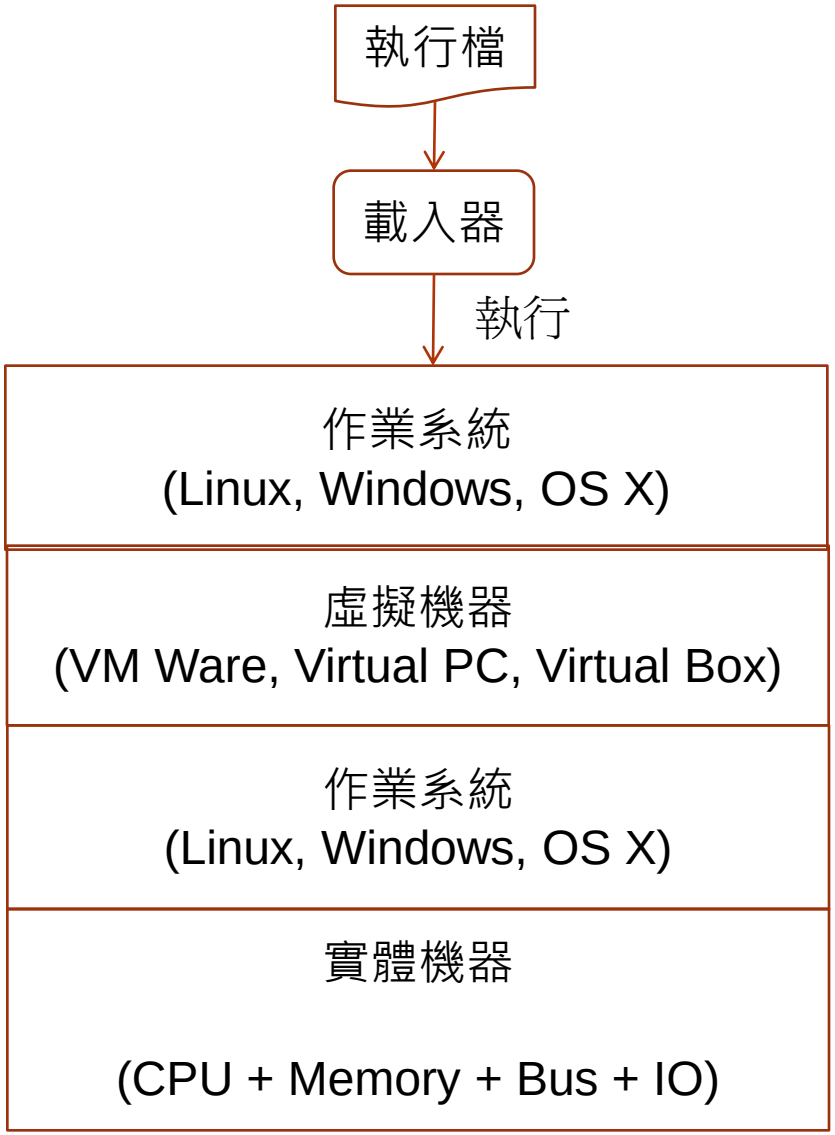
(b) 系統式虛擬機

(a) 無虛擬機時

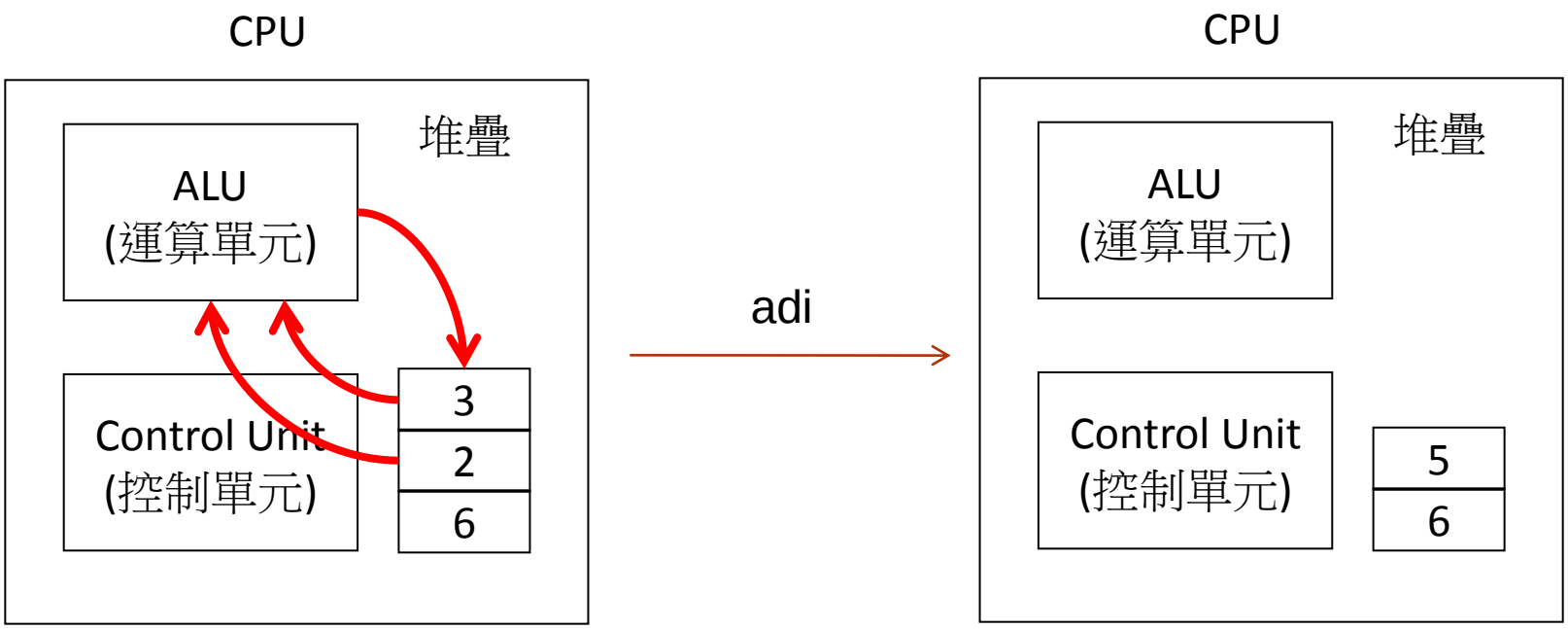
(b) 原生式虛擬機

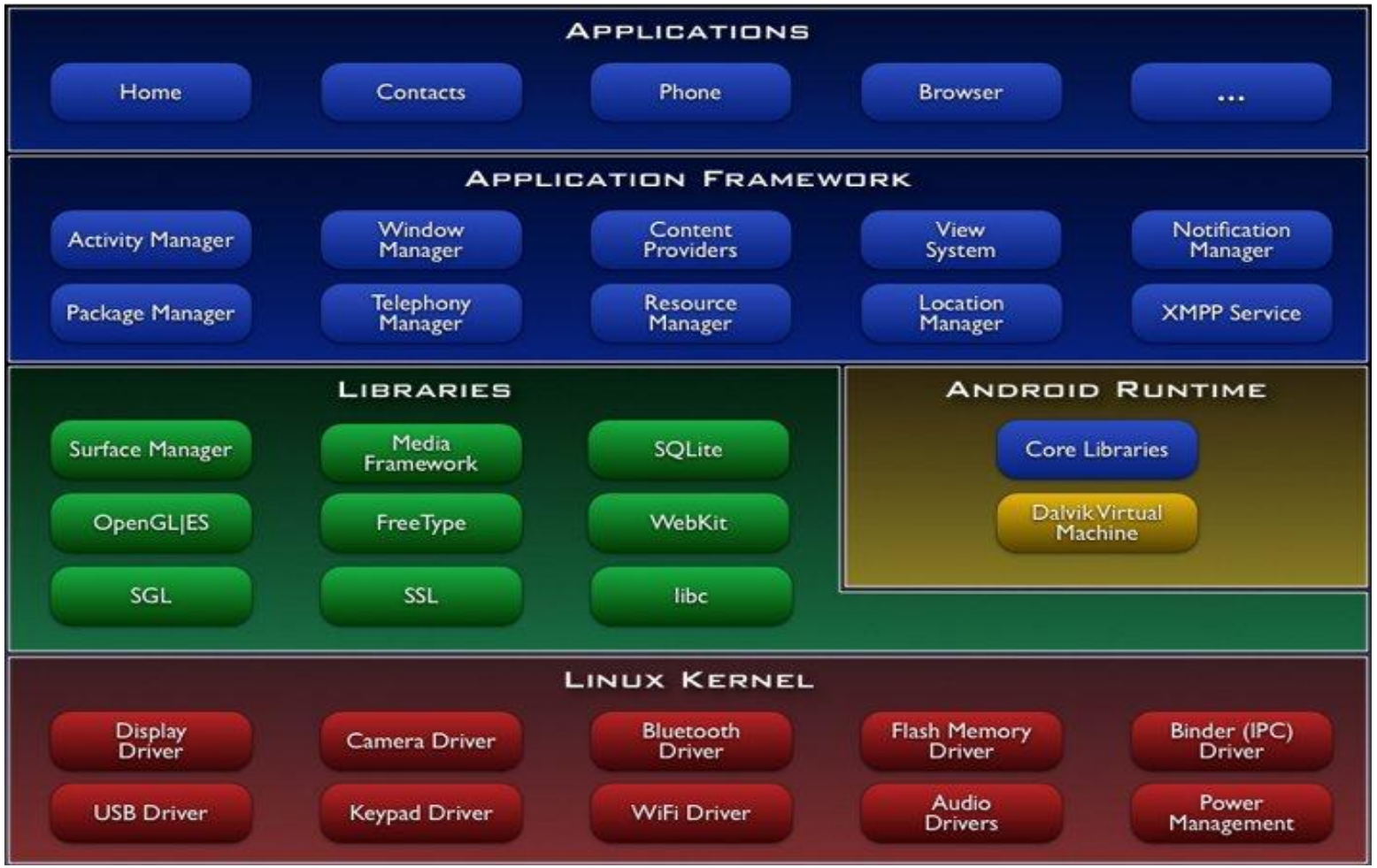


(c) 寄生式虛擬機

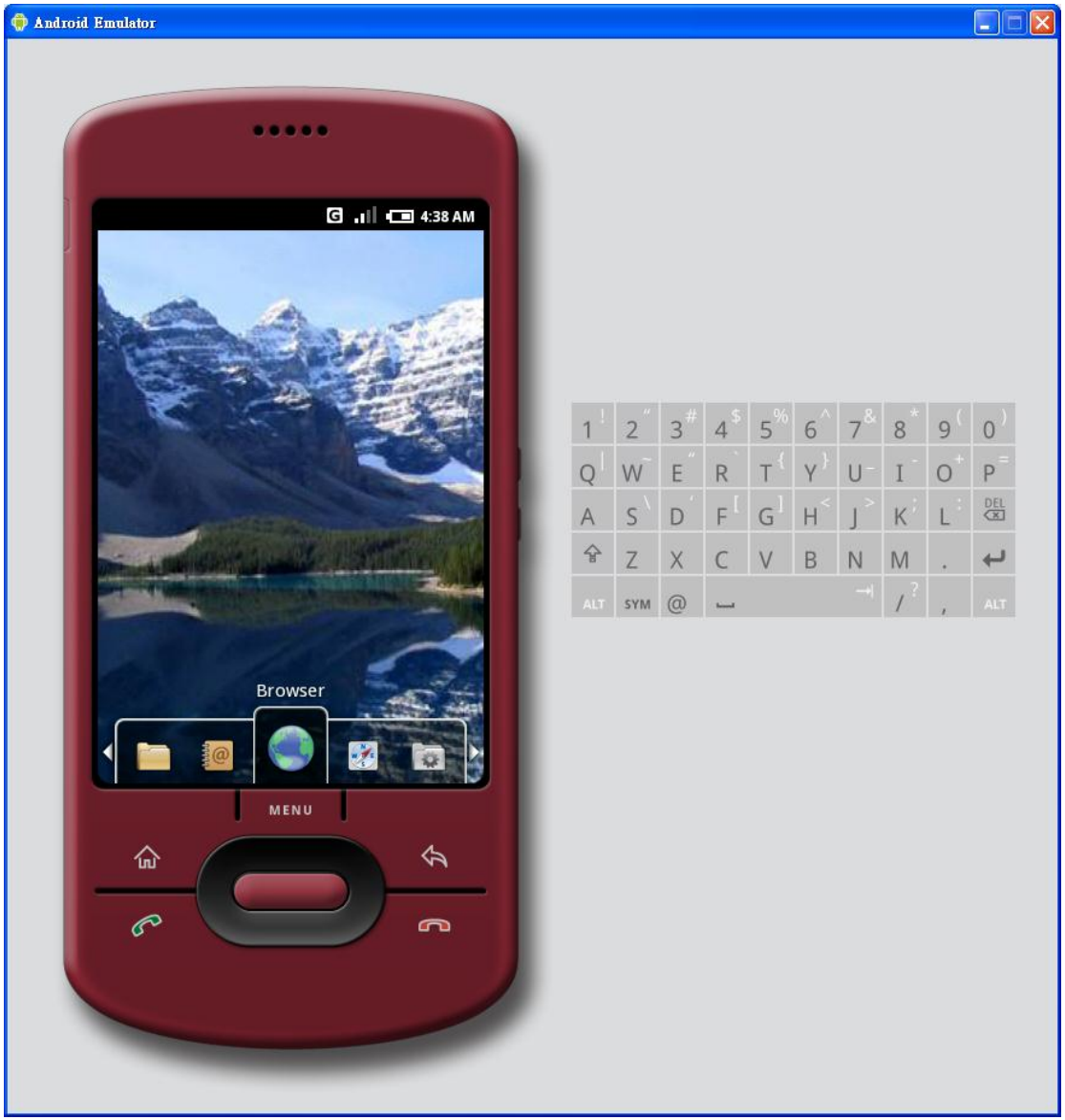

執行檔
載入器
執行
作業系統
(Linux, Windows, OS X)
虛擬機器
(VM Ware, Virtual PC, Virtual Box)
作業系統
(Linux, Windows, OS X)
實體機器
(CPU + Memory + Bus + IO)

堆疊機的 adi 指令之執行過程示意圖




Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
XMPP Service
Libraries
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Google 平台Android的模擬器環境



微軟 Visual Studio 中的智慧型手機模擬器