

編譯程式設計

期末專題說明 V1.1

April 2003

專題目標

- 實作一個可用的編譯器
- 編譯特定的程式語言並產生可以在IA-32架構上執行的組合語言及可執行檔
- 編譯器與產生的執行檔必須可以在 Linux 2.4 作業系統下執行
- 使用NASM作為組譯器
- 可和libc連結

2

編譯器的輸出入格式

- 請參見老師的投影片

3

錯誤訊息格式

- 檔案名稱:行號:訊息內容
 - test1.p:10:array index out of range
 - test2.p:8:incompatible variable type assignment

4

程式語言

- 命令式語言
- 由最簡單到最難共分為四個階段
 - 依完成度作為評分依據

5

基本語法

```
#  
# Program 1  
#  
program  
begin  
...  
end
```

- 序述以換行符號作為結尾
- 標記間可以有任意個數的空白
- 符號'#'開始到行尾為註解
- 程式由 `program` 宣告開始
- 程式主體寫在 `begin` 和 `end` 之間
- 大小寫視為不同

6

第一階段 (1/3)

```
var  
  var1, var2, ...: int
```

■ 變數宣告

- 接在 program 宣告之後, begin 之前, 可有可無
- 變數宣告區段以 var 開頭, 可以有多行
- 可宣告一個或多個變數, 以逗號分開
- 變數名稱最長不會超過 255 個字元
- 變數名稱可以是英文字母大小寫, 數字或是底線, 但是開頭不可以是數字
- 冒號之後為變數型態, 只需要支援 int
 - 32位元有號整數型態

7

第一階段 (2/3)

```
x := 1  
y := 2  
z := (x + y) * (x - y) # z 的值为 -3
```

■ 變數值設定

- :=

■ 基本整數四則運算

- +, -, *, /, mod, (), -(負號)

8

第一階段 (3/3)

```
x := read()  
y := read()  
write(x + y)  
writeln()
```

■ 整數的輸入與輸出

- 呼叫 read() 可以由鍵盤輸入一整數並傳回
- 呼叫 write(x) 將 x 的內容以十進位方式輸出到螢幕上, 並在最後附加一空白字元
- writeln() 輸出換行符號到螢幕上

9

第一階段的程式範例

```
#  
# Level 1 Program  
#  
program  
var  
  i, j, k: int  
begin  
  i := read()  
  j := read()  
  k := (i + j) * (i - j)  
  write(k)  
  writeln()  
end
```

10

第二階段 (1/2)

```
(x >= y) and not (x <= z)
```

■ 比較運算

- >, >=, <, <=, =

■ 布林運算

- and, or, not

■ 零值為假, 非零值為真

11

第二階段 (2/2)

```
while(...) do  
  ...  
endwhile  
  
if(...) then  
  ...  
else  
  # 這部份可有可無  
  ...  
endif
```

■ if-then-else

- else的部份可有可無

■ while 迴圈控制

■ 皆允許巢狀結構

12

第二階段的程式範例

```
#
# Level 2 Program
#
program
var
  i, result, x: int
begin
  x := read()
  result := 0
  i := 1
  while(i <= x) do
    result := result + i
    i := i + 1
  endwhile
  write(result)
  writeln()
end
```

13

第三階段 (1/3)

```
var
  a, b, c: char
  x, y, z: int
begin
  a := 'a'
  x := 1
  y := a # 這行會造成編譯錯誤
end
```

- 字元和整數資料型態
 - 編譯時能夠檢查型態是否相符
 - 字元常數用單引號括起來

14

第三階段 (2/3)

```
var
  a, b: int[2]
begin
  a[0] := 1
  a[1] := 2
  a[2] := 3 # 這行會造成編譯錯誤
  b := a
end
```

- 字元陣列與整數陣列
 - 索引值從0開始
 - 用方括號定址
 - 編譯時與程式執行時都必須能夠偵測陣列索引值是否超過範圍，並印出錯誤訊息
 - 允許陣列拷貝，複製整個陣列的內容，如果來源陣列比目的陣列大的話則印出錯誤訊息

15

第三階段 (3/3)

```
var
  str: char[6]
begin
  str := "hello"
  writestr(str)
end
```

- 字串與字串的輸出
 - 字串使用字元陣列表示，以'\$'字元結尾
 - 字串常數使用雙引號括起來，並在結尾自動附加'\$'字元
 - 字串使用writestr()輸出到螢幕上

16

第三階段的程式範例

```
#
# Level 3 Program
#
program
var
  a: int[2]
begin
  a[0] := read()
  a[1] := read()
  writestr("The first integer is ")
  write(a[0])
  writeln()
  writestr("The second integer is ")
  writeln()
  write(a[1])
end
```

17

第四階段

```
sub fname(a, b: int; x: char: ...)
var
  local1, local2, ...: int # 可有可無
begin
  ...
end
```

- 函式宣告
 - 宣告在全域變數之後，主程式之前
 - 不需支援forward reference
 - 允許遞迴呼叫
 - Call by Value
 - 用分號格開不同型態的傳入參數

18

第四階段的程式範例

```
# Level 4 Program
#
program
var
  str: char[6]

sub hello()
begin
  writestr(str)
  writeln()
end

# 底下是主程式
str := "Hello"
hello()
end
```

19

還想要更多?

- 其它自己想加的功能
 - Error Handling
 - Pointer, and Pointer Arithmetic
 - Call by Reference, Call by Address
 - User-defined Data Structure
 - Multi-dimension Array
 - Nested Function Definition
 - Time/Space Optimization
 - *Everything but user interface.....*
- 可以自行擴充語言文法，但務必要能處理之前的各項要求

20

附錄 - 保留字

- 保留字不可以用作變數或函式名稱
- and, begin, char, do, else, end, endif, endwhile, if, int, mod, not, or, program, sub, then, var, while

21

附錄 - 運算元結合優先順序

- 優先順序由上到下遞減，結合方向由左至右
 - () []
 - not, -(負號)
 - * / mod
 - + -
 - < <= > >=
 - =
 - and
 - or

22