# 程式人 月刊 雜誌

# Programmer



讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌

參考價： NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體

羅慧夫顱顏基金會 彰化銀行 (009) 帳號： 5234-01-41778-800

*0123778*

愛心條碼

# 程式人雜誌

2015 年 9 月

本期焦點：向 Nand2Tetris 學習系統軟體設計

# 程式人雜誌


- 前言
  - 編輯小語
  - 授權聲明
- 本期焦點：向 Nand2Tetris 學習系統軟體設計
  - Nand2Tetris 第 6 週 -- 自製組譯器
  - Nand2Tetris 第 7-8 週 -- 中間碼轉組合語言
  - Nand2Tetris 第 9-11 週 -- 編譯器
  - Nand2Tetris 第 12 週 -- 作業系統
- 雜誌訊息
  - 讀者訂閱
  - 投稿須知
  - 參與編輯
  - 公益資訊

# 前言

## 編輯小語

最近小編發現了一個很棒的網路公開課，是 coursera 課程網站上的 From Nand to Tetris / Part I 這門課，於是我去修了這門課並作了習題，這是我真正修習的第一門網路公開課。

這門課程是教授學生如何從一個最基礎的 nand 邏輯閘開始，一路往上建構出所有基礎元件，像是 and, or, xor, not, MUX, DMUX, Adder, Memory 等等，接著建構出 CPU 與整台電腦，然後再學習如何在建構出這台電腦上的『組譯器、編譯器、作業系統』等等，最後在這個具備軟硬體的電腦上寫一個小型的方塊遊戲。

我覺得這門課非常棒，因此小編將整個修課的過程與心得分享給大家，也利用這個機會將這門『神級課程』介紹給大家認識。

----（「少年科技人雜誌」與「程式人雜誌」編輯 - 陳鍾誠）

## 授權聲明

# 本期焦點：向 **Nand2Tetris** 學習系統軟體設計

## **Nand2Tetris** 第 **6** 週 **--** 自製組譯器

本文程式修改自下列 github 網址，使用時請注意遵守開源授權。

- https://github.com/havivha/Nand2Tetris/tree/master/6

關於 nand2tetris 第 6 週的內容與解說，請參考下列文章。

- 少年科技人雜誌 / 2015年6月號 / Nand2Tetris 第 6 週 -- 組譯器

在本文中，我們主要補充的是實作的方法。我們採用 JavaScript 撰寫，並用 node.js 環境進行測試。

由於 HackCPU 的獨特設計方式，讓組譯器牽涉到比較多的表格，以下是這些表格的內容。

```
var dtable = {
  ""    :0b000,
```

```
  "M"  :0b001,
  "D"  :0b010,
  "MD" :0b011,
  "A"  :0b100,
  "AM" :0b101,
  "AD" :0b110,
  "AMD":0b111
}

var jtable = {
  ""   :0b000,
  "JGT":0b001,
  "JEQ":0b010,
  "JGE":0b011,
  "JLT":0b100,
  "JNE":0b101,
```

```
  "JLE":0b110,
  "JMP":0b111
}

var ctable = {
  "0"   :0b0101010,
  "1"   :0b0111111,
  "-1"  :0b0111010,
  "D"   :0b0001100,
  "A"   :0b0110000,
  "M"   :0b1110000,
  "!D"  :0b0001101,
  "!A"  :0b0110001,
  "!M"  :0b1110001,
  "-D"  :0b0001111,
  "-A"  :0b0110011,
```

```
    "-M"  :0b1110011,
    "D+1" :0b0011111,
    "A+1" :0b0110111,
    "M+1" :0b1110111,
    "D-1" :0b0001110,
    "A-1" :0b0110010,
    "M-1" :0b1110010,
    "D+A" :0b0000010,
    "D+M" :0b1000010,
    "D-A" :0b0010011,
    "D-M" :0b1010011,
    "A-D" :0b0000111,
    "M-D" :0b1000111,
    "D&A" :0b0000000,
    "D&M" :0b1000000,
    "D|A" :0b0010101,
```

```
  "D|M" :0b1010101
}

var symTable = {
  "R0"  :0,
  "R1"  :1,
  "R2"  :2,
  "R3"  :3,
  "R4"  :4,
  "R5"  :5,
  "R6"  :6,
  "R7"  :7,
  "R8"  :8,
  "R9"  :9,
  "R10" :10,
  "R11" :11,
```

```
  "R12" :12,
  "R13" :13,
  "R14" :14,
  "R15" :15,
  "SP"  :0,
  "LCL" :1,
  "ARG" :2,
  "THIS":3,
  "THAT":4,
  "KBD" :24576,
  "SCREEN":16384
};
```

然後和傳統的組譯器一樣，我們可以採用兩階段的組譯方式，第一階段紀錄每個符號的位址，第二階段則進行編碼的動作。

```
function pass1(lines) {
```

```
  // 對於每一行程式碼
  //   計算該行的位址
  //   如果有符號，紀錄符號的位址
}

function pass2(lines, objFile) {
  // 對於每一行程式碼
  //   將指令轉為機器碼（查表，編碼，輸出）
}

// 轉換指令為程式碼的主要程式如下。
function toCode(p) {
  var address;
  if (p.type === "A") {
    if (p.arg.match(/^\d+$/)) {
      address = parseInt(p.arg);
```

```
    } else {
      address = symTable[p.arg];
      if (typeof address === 'undefined') {
        address = symTop;
        addSymbol(p.arg, address);
      }
    }
    return address;
  } else { // if (p.type === "C")
    var d = dtable[p.d];
    var cx = ctable[p.c];
    var j = jtable[p.j];
    return 0b111<<13|cx<<6|d<<3|j;
  }
}
```

只要搞清楚指令格式，並且把表格寫出來， HackCPU 的組譯器算是相當容易撰寫的一個程式。

## Nand2Tetris 第 7-8 週 -- 中間碼轉組合語言

本文程式修改自下列 github 網址，使用時請注意遵守開源授權。

- https://github.com/havivha/Nand2Tetris/tree/master/7
- https://github.com/havivha/Nand2Tetris/tree/master/8

關於 nand2tetris 第 7-8 週的內容與解說，請參考下列文章。

- 少年科技人雜誌 / 2015年8月號 / Nand2Tetris 第 7 週 -- VM I: Stack Arithmetic
- 少年科技人雜誌 / 2015年8月號 / Nand2Tetris 第 8 週 -- VM II: Program Control

在本文中，我們主要補充的是實作的方法。我們採用 JavaScript 撰寫，並用 node.js 環境進行測試。

以下是筆者 Nand2Tetris 第 7-8 週習題實作中，程式的主要架構部分。

```
function vm2asm(vmFile) {
  ...
  pass1(lines);
}

function pass1(lines, asmFile) {
  c.log("============== pass1 ================");
  for (var i=0; i<lines.length; i++) {
    var p = parse(lines[i], i);
    if (p===null) continue;
    comment(lines[i]);
    translate(p);
  }
}
```

```
function parse(line, i) {
  ...
}

function translate(p) {
  var p1 = p.tokens[1];
  var p2 = p.tokens[2];
  switch (p.op) {
    case "push": cmdPush(p1, parseInt(p2)); break;
    case "pop":  cmdPop(p1, parseInt(p2)); break;
    case "add":  cmdBinary("D+A"); break;
    case "sub":  cmdBinary("A-D"); break;
    case "and":  cmdBinary("D&A"); break;
    case "or" :  cmdBinary("D|A"); break;
    case "not":  cmdUnary("!D"); break;
    case "neg":  cmdUnary("-D"); break;
```

```
    case "eq" :  cmdCompare("JEQ"); break;
    case "gt" :  cmdCompare("JGT"); break;
    case "lt" :  cmdCompare("JLT"); break;
    case "goto": cmdGoto(p1); break;
    case "if-goto": cmdIfGoto(p1); break;
    case "label": cmdLabel(p1); break;
    case "function": cmdFunction(p1, parseInt(p2)); break;
    case "call"  : cmdCall(p1, parseInt(p2)); break;
    case "return": cmdReturn(); break;
    default: error();
  }
}
```

接著就是每個指令轉換的實作部份，為了避免公佈解答，我們僅顯示部份程式碼，以便讓讀者知道整體架構。

```
function cmdGoto(label) {
```

```
  A(label);
  C("0;JMP");
}

function cmdIfGoto(label) {
  popToDest('D');
  A(label);
  C('D;JNE');
}

function cmdUnary(comp) {
  decSp();                  // --SP
  stackToDest('D');         // D=*SP
  C('D='+comp);             // D=COMP
  compToStack('D');         // *SP=D
  incSp();                  // ++SP
```

```
}

function cmdCompare(j) {
  decSp();                // --SP
  stackToDest('D');       // D=*SP
  decSp();                // --SP
  stackToDest('A');       // A=*SP
  C('D=A-D');             // D=A-D
  var label_eq = jump('D', j);  // D;jump to label_eq
  compToStack('0');       // *SP=0
  var label_ne = jump('0', 'JMP'); // 0;JMP to label_ne
  cmdLabel(label_eq);     // (label_eq)
  compToStack('-1');      // *SP=-1
  cmdLabel(label_ne);        // (label_ne)
  incSp();                // ++SP
}
```

```
function cmdBinary(comp) {
  decSp();            // --SP
  stackToDest('D');   // D=*SP
  decSp();            // --SP
  stackToDest('A');   // A=*SP
  C('D='+comp);       // D=comp
  compToStack('D');   // *SP=D
  incSp();
}

function cmdLabel(name) {
  asm("("+name+")");
}

function cmdFunction(name, n) {
```

```
  cmdLabel(name);
  for (var i=0; i<n; i++) {
    cmdPush("constant", 0);
  }
}

function cmdCall(name, n) {
  var returnAddress = newLabel();
  cmdPush("constant", returnAddress); // push return_address
  cmdPush("reg", R_LCL);              // push LCL
  cmdPush("reg", R_ARG);              // push ARG
  cmdPush("reg", R_THIS);             // push THIS
  cmdPush("reg", R_THAT);             // push THAT
  loadSeg("R"+R_SP, -n-5, true);
  compToReg(R_ARG, 'D');              // ARG=SP-n-5
  regToReg(R_LCL, R_SP);              // LCL=SP
```

```
  A(name);                              // A=function_name
  C('0;JMP');                           // 0;JMP
  cmdLabel(returnAddress);              // (return_address)
}

function regToReg(dest, src) {
  regToDest('D', src);
  compToReg(dest, 'D');          // Rdest = Rsrc
}

function prevFrameToReg(reg) {
  regToDest('D', R_FRAME);       // D=FRAME
  C('D=D-1');                    // D=FRAME-1
  compToReg(R_FRAME, 'D');       // FRAME=FRAME-1
  C('A=D');                      // A=FRAME-1
  C('D=M');                      // D=*(FRAME-1)
```

```
  compToReg(reg, 'D');          // reg=D
}

function cmdReturn() {
  regToReg(R_FRAME, R_LCL);    // R_FRAME = R_LCL
  A("5");                      // A=5 ??
  C("A=D-A");                  // A=FRAME-5
  C("D=M");                    // D=M
  compToReg(R_RET, 'D');       // RET=*(FRAME-5)
  cmdPop("argument", 0);       // *ARG=return value
  regToDest('D', R_ARG);       // D=ARG
  compToReg(R_SP, 'D+1');      // SP=ARG+1
  prevFrameToReg(R_THAT);      // THAT=*(FRAME-1)
  prevFrameToReg(R_THIS);      // THIS=*(FRAME-2)
  prevFrameToReg(R_ARG);       // ARG=*(FRAME-3)
  prevFrameToReg(R_LCL);       // LCL=*(FRAME-4)
```

```
  regToDest('A', R_RET);       // A=RET
  C("0;JMP");                  // goto RET
}

function jump(comp, j) {
  var label = newLabel();
  A(label);
  C(""+comp+";"+j);
  return label;
}

function init() {
  comment("init");
  A('256');
  C('D=A');
  compToReg(R_SP, 'D');     // SP=256
```

```
  cmdCall('Sys.init', 0);  // call Sys.init()
}
```

當您了解了上述的主架構之後，應該就能開始自己重新打造完整的程式了。

## Nand2Tetris 第 9-11 週 -- 編譯器

本文程式修改自下列 github 網址，使用時請注意遵守開源授權。

- https://github.com/havivha/Nand2Tetris/tree/master/10
- https://github.com/havivha/Nand2Tetris/tree/master/11

關於 nand2tetris 第 9-11 週的內容與解說，請參考下列文章。

- 少年科技人雜誌 / 2015年8月號 / Nand2Tetris 第 9 週 -- High-Level Language
- 少年科技人雜誌 / 2015年8月號 /Nand2Tetris 第 10 週 -- Compiler I: Syntax Analysis
- 少年科技人雜誌 / 2015年8月號 / Nand2Tetris 第 11 週 -- Compiler II: Code Generation

在本文中，我們主要補充的是實作的方法。我們採用 JavaScript 撰寫，並用 node.js 環境進行測試。

我們的編譯器主要分成兩個部分，詞彙切分 (Lexer) 和語法剖析 (Parser) ，然後當語法剖析完成後，我們在 Parser 當中插入產生中間碼的程式，將 Parser 擴充為一個編譯器 Compiler 。

為了避免公布解答，我們在此僅列出部分程式碼以供參考!

## 詞彙切分 **(Lexer)**

```
var lexer = {
  tokens : [],
  tokenIdx : 0
}

lexer.scanFile=function(path) {
  // 讀取 *.jack 程式檔，並且切分成一個一個的詞彙放入 tokens 陣列中。
  ...
  var text = fs.readFileSync(path, "utf8");
```

```
  ...
  // 接著用一個 regular expression 就能進行詞彙切分的動作。
  var re = /\/\*([\s\S]*?)\*\/|\/\/([^\r\n]*)[\r\n]|"(.*?)"|(\d+)
|([a-zA-Z]\w*)|([>=<!\+\-\*\/&%|\.\(\)\{\}\[\];,])|(\s+)|(.)/gm;
  var types = [ "", "comment", "comment", "stringConstant", "inte
gerConstant", "identifier", "symbol", "space", "ch" ];
  ...
  tokens = [];
  tokenIdx = 0;
  ...
  while((m = re.exec(text)) !== null) {
    ...
  }
  return tokens;
}
...
```

語法剖析 **(Parser)**

```
var compileFile=function(path) {
  ...
  tokens = lexer.scanFile(path);
  ...
  Class();
  ...
}

// class : 'class' VarName '{' classVarDec* subroutineDec* '}'
var Class=function() {
  pushNT("class");
  next("class");
  varKind = "class";
```

```
  className = VarName();
  classStaticList=[]; classFieldList=[]; classSubroutineList=[];
classSymTable={};
  next("{");
  while (isNext("static") || isNext("field")) {
    ClassVarDec();
  }
  while (!isNext('}')) {
    SubroutineDec();
  }
  popNT("class");
  console.log("classStaticList=%j", classStaticList);
  console.log("classFieldList=%j", classFieldList);
  console.log("classSubroutineList=%j", classSubroutineList);
}
```

```
// classVarDec: {'static'|'field'} type varList ';'
var ClassVarDec=function() {
  pushNT("classVarDec");
  varKind = next(); // {'static'|'field'}
  Type();
  VarList();
  next(";");
  popNT("classVarDec");
}

// varList: varName (',' varName)*
var VarList=function() {
  var list = [];
  var name=VarName();
  while (isNext(",")) {
    next(",");
```

```
    name = VarName();
  }
}

// subroutineDec: ('constructor'|'function'|'method') ('void'|typ
e) varName '(' parameterList? ')' subroutineBody
var SubroutineDec=function() {
  pushNT("subroutineDec");
  ...
  var tag = next(/(constructor)|(function)|(method)/); // ('const
ructor'|'function'|'method')
  if (isType() || isNext("void")) {
    var vtype = next();
    ...
    next("(");
    if (!isNext(")")) {
```

```
      ParameterList();
    }
    next(")");
    SubroutineBody();
  }
  popNT("subroutineDec");
}

// parameterList: (parameter (',' parameter)*)?
var ParameterList=function() {
  pushNT("parameterList");
  Parameter();
  while (isNext(",")) {
    next(",");
    Parameter();
  }
```

```
  popNT("parameterList");
}

...

// subroutineBody: '{' varDec* statements '}'
var SubroutineBody=function() {
  pushNT("subroutineBody");
  next("{");
  while (isNext("var")) {
    VarDec();
  }
  Statements();
  next("}");
  popNT("subroutineBody");
}
```

```
// varDec: 'var' type varList ';'
var VarDec=function() {
  pushNT("varDec");
  next("var");
  varKind = "var";
  Type();
  VarList();
  next(";");
  popNT("varDec");
}

// statements : statement*
var Statements=function() {
  pushNT("statements");
  while (!isNext("}")) {
```

```
    Statement();
  }
  popNT("statements");
}

// statement: letStatement | ifStatement | whileStatement | doSta
tement | returnStatement
var Statement=function() {
  if (isNext("let")) {
    LetStatement();
  } else if (isNext("if")) {
    IfStatement();
  } else if (isNext("while")) {
    WhileStatement();
  } else if (isNext("do")) {
    DoStatement();
```

```
  } else if (isNext("return")) {
    ReturnStatement();
  } else {
    error("STATEMENT=>");
  }
}

// letStatement: 'let' varName ( arraySubscript )? '=' expression
 ';'
var LetStatement=function() {
  pushNT("letStatement");
  next("let");
  ...
  VarName();
  if (isNext("[")) {
    ArraySubscript();
```

```
  }
  next("=");
  Expression();
  next(";");
  popNT("letStatement");
}

// arraySubscript: '[' expression ']'
var ArraySubscript=function() {
  next("[");
  Expression();
  next("]");
}

// block = '{' statements '}'
var Block=function() {
```

```
  next("{");
  Statements();
  next("}");
}

// ifStatement : 'if' '(' expression ')' block ('else' block)?
var IfStatement=function() {
  pushNT("ifStatement");
  next("if");
  next("(");
  Expression();
  next(")");
  Block();
  if (isNext("else")) {
    next("else");
    Block();
```

```
  }
  popNT("ifStatement");
}

// whileStatement: 'while' '(' expression ')' block
var WhileStatement=function() {
  pushNT("whileStatement");
  next("while");
  next("(");
  Expression();
  next(")");
  Block();
  popNT("whileStatement");
}

// doStatement: 'do' call ';'
```

```
var DoStatement=function() {
  pushNT("doStatement");
  next("do");
  Call();
  next(";");
  popNT("doStatement");
}

// returnStatement: 'return' expression? ';'
var ReturnStatement=function() {
  pushNT("returnStatement");
  next("return");
  if (!isNext(";"))
    Expression();
  next(";")
  popNT("returnStatement");
```

```
}

// expression: term (op term)*
var Expression=function() {
  pushNT("expression");
  Term();
  while (isNext(/[\+\-\*\/\&<>=]/)) { // op in "+/-*&<>="
    var op = next();
    Term();
  }
  popNT("expression");
}

...

// call: subroutineName '(' expressionList ')' | (className | var
```

```
Name) '.' subroutineName '(' expressionList ')'
var Call=function() {
  var name = nextType(ID);
  if (isNext("(")) {
    next("(");
    ExpressionList();
    next(")");
  } else if (isNext(".")) {
    next(".");
    var subName = nextType(ID);
    next("(")
    ExpressionList();
    next(")");
  }
}
```

```
// expressionList: (expression (',' expression)*)?
var ExpressionList=function() {
  pushNT("expressionList");
  Expression();
  while (isNext(",")) {
    next(",");
    Expression();
  }
  popNT("expressionList");
}
```

## 中間碼產生 **(Code Generation)**

產生中間碼的動作直接嵌入在語法剖析動作中執行，例如以下的程式碼中，有 gen.command(...) 之類的動作就是用來產生中間碼的。

```
// subroutineBody: '{' varDec* statements '}'
```

```
var SubroutineBody=function() {
  pushNT("subroutineBody");
  next("{");
  varCount = 0;
  while (isNext("var")) {
    VarDec();
  }
  gen.method(methodTag, className+"."+methodName, varCount);
  if (methodTag === "method") { // method 函數預設第一個參數為 th
is
    gen.command('push argument 0');
    gen.command('pop pointer '+POINTER_THIS); // set up 'this' po
inter
  } else if (methodTag === "constructor") {
    genPush(classFieldList.length);
    gen.command('call Memory.alloc 1');
```

```
    gen.command('pop pointer '+POINTER_THIS); // set up 'this' po
inter
  }
  Statements();
  next("}");
  popNT("subroutineBody");
}

// varName: ID
var VarName=function() {
  var name=nextType(ID);
  if (varKind == "let") return name;
  var symbol = { name:name, type:varType, kind:varKind };
  switch (varKind) {
    case "class": break;
    case "static": addSymbol(classSymTable, classStaticList, symb
```

```
ol); break;
    case "field": symbol.kind="this"; addSymbol(classSymTable, cl
assFieldList, symbol); break;
    case "subroutine":addSymbol(classSymTable, classSubroutineLis
t, symbol); break;
    case "argument": addSymbol(methodSymTable, methodArgList, sym
bol); break;
    case "var": symbol.kind="local"; addSymbol(methodSymTable, me
thodVarList, symbol); break;
    default:
      error("VarKind not Found!");
  }
  return name;
}

// arraySubscript: '[' expression ']'
```

```
var ArraySubscript=function(arrayName) {
  genPush(arrayName);
  next("[");
  Expression();
  next("]");
  gen.command("add");
  gen.command("pop pointer "+POINTER_THAT); // pop into that ptr
  gen.command("push that 0"); //  push *(base+index) onto stack
}

// condExpression = ( expression ) block
var CondExpression=function(gotoLabel) {
  next("(");
  Expression();
  next(")");
  gen.command("not"); // ~(cond)
```

```
  var notifLabel = newLabel();
  gen.command("if-goto "+notifLabel);
  Block();
  gen.command("goto "+gotoLabel);
  gen.label(notifLabel);
}
```

而 gen 這個用來輸出中間碼的物件，其部分程式碼如下：

```
gen.stringConstant=function(str) {
  command('push constant '+str.length);
  gen.call("String.new", 1);
  for (var i=0; i<str.length; i++) {
    command('push constant '+str.charCodeAt(i));
    gen.call("String.appendChar", 2);
  }
```

```
}

gen.keywordConstant=function(str) {
  switch (str) {
    case "this" : gen.command("push pointer 0"); break;
    case "true" : gen.command("push constant 1"); gen.command("ne
g"); break;
    case "false": gen.command("push constant 0"); break;
    case "null" : gen.command("push constant 0"); break;
    default: throw Error("keywordConstant not found :"+str);
  }
}

gen.command = function(cmd) {
  command(cmd);
}
```

以上用程式碼片段說明了我們的編譯器實作方法，您可以看到這個編譯器的設計方式採用的是遞迴下降法，也就是直接用遞迴程式剖析 EBNF 語法的方式，來處理 JACK 這個程式語言的語法。

## Nand2Tetris 第 12 週 -- 作業系統

本文程式來自 https://github.com/havivha/Nand2Tetris/tree/master/12 ，使用時請注意遵守開源授權。

關於 nand2tetris 第 12 週的內容與解說，請參考下列文章。

- 少年科技人雜誌 / 2015年8月號 / Nand2Tetris 第 12 週 -- Operating System

在本文中，我們主要補充的是實作的方法。我們採用 JavaScript 撰寫，並用 node.js 環境進行測試。 檔案: Array.jack

```
// Represents an array. Can be used to hold any type of object.
class Array {
```

```
    // Constructs a new Array of the given size.
    function Array new(int size) {
        return Memory.alloc(size);
    }

    // De-allocates the array and frees its space.
    method void dispose() {
        do Memory.deAlloc(this);
        return;
    }
}
```

檔案: Memory.jack

```
class Memory {
    static Array memory;
```

```
    static Array freeList;
    static Array NO_BLOCK;

    // Free block structure:
    // word 0: free block size including 2 header words
    // word 1: Next free block ptr
    static int FL_LENGTH; // freeList.length index
    static int FL_NEXT;   // freeList.next index

    // Alloc block structure:
    // word 0: alloc block size including 1 header word
    // word 1..size: allocated block
    static int ALLOC_SIZE;// alloc block size index relative to s
tart of allocated block

    /** Initializes memory parameters. */
```

```
    function void init() {
        let memory = 0;
        let freeList = 2048;
        let NO_BLOCK = 16384;   // means no block found
        let FL_LENGTH = 0;
        let FL_NEXT = 1;
        let ALLOC_SIZE = -1;
        let freeList[FL_LENGTH] = 16384-2048;
        let freeList[FL_NEXT] = null;
        return;
    }

    /** Returns the value of the main memory at the given address
. */
    function int peek(int address) {
        return memory[address];
```

```
    }

    /** Sets the value of the main memory at this address
     *  to the given value. */
    function void poke(int address, int value) {
        let memory[address] = value;
        return;
    }

    /** finds and allocates from the heap a memory block of the
     *  specified size and returns a reference to its base addres
s. */
    function Array alloc(int size) {
        var Array prev_block;
        var Array found_block;
```

```
        let prev_block = Memory.best_fit(size);              // pr
ev_block is the block before the found block
        if( prev_block = NO_BLOCK ) {
            let found_block = null;    // none found
        }
        else {
            if( prev_block = null ) {
                let found_block = freeList;                  // Ne
w block is at the beginning of the freeList
                let freeList = Memory.do_alloc(found_block, size)
; // Free list now starts a new first free block
            }
            else {
                let found_block = prev_block[FL_NEXT];
                let prev_block[FL_NEXT] = Memory.do_alloc(found_b
lock, size);
```

```
            }
        }
        return found_block+1;
    }
    ....
}
```

檔案: String.jack

```
class String {
  constructor String new(int maxLength)
  method void dispose()
  method int length()
  method char charAt(int j)
  method void setCharAt(int j, char c)
  method String appendChar(char c)
```

```
  method void eraseLastChar()
  method int intValue()
  method void setInt(int j)
  function char backSpace()
  function char doubleQuote()
  function char newLine()
}
```

檔案: Screen.jack

```
class Screen {
    static Array screen;
    static boolean cur_colour;
    static int white_pixel;
    static int black_pixel;
    static boolean white;
```

```
    static boolean black;

    /** Initializes the Screen. */
    function void init() {
        let screen = 16384;
        let white = false;
        let black = true;
        let white_pixel = 0;
        let black_pixel = 1;
        let cur_colour = black;
        return;
    }


    /** Erases the whole screen. */
    function void clearScreen() {
```

```
        var int i;
        let i = 0;
        while( i < 8192 ) {
            let screen[i] = white;
        }
        return;
    }

    /** Sets the color to be used in further draw commands
     *  where white = false, black = true. */
    function void setColor(boolean b) {
        let cur_colour = b;
        return;
    }

function void clearScreen()
```

```
  function void setColor(boolean b)
  function void drawPixel(int x, int y)
  function void drawLine(int x1, int y1, int x2, int y2)
  function void drawRectangle(int x1, int y1,int x2, int y2)
  function void drawCircle(int x, int y, int r)
}
```

檔案: Output.jack

```
class Output {
    /** Initializes the screen and locates the cursor at the scre
en's top-left. */
    function void init() {
        let screen = 16384;
        let cursor_x = 0;
        let cursor_y = 0;
```

```
        let charMasks = Array.new(2);
        let charMasks[0] = 255;
        let charMasks[1] = -1 & 255;
        do Output.initMap();
        return;
    }

    // Initalizes the character map array
    function void initMap() {
        var int i;

        let charMaps = Array.new(127);

        // black square (used for non printable characters)
        do Output.create(0,63,63,63,63,63,63,63,63,63,0,0);
```

```
        // Assigns the bitmap for each character in the charachte
r set.
        do Output.create(32,0,0,0,0,0,0,0,0,0,0,0);           //
        do Output.create(33,12,30,30,30,12,12,0,12,12,0,0);  // !
        do Output.create(34,54,54,20,0,0,0,0,0,0,0,0);        // "
        do Output.create(35,0,18,18,63,18,18,63,18,18,0,0);  // #
...
  function void moveCursor(int i, int j)
  function void printChar(char c)
  function void printString(String s)
  function void printInt(int i)
  function void println()
  function void backSpace()
}
```

檔案: Keyboard.jack

```
class Keyboard {
    static Array keyboard;

    /** Initializes the keyboard. */
    function void init() {
        let keyboard = 24576;
        return;
    }
    /**
     * Reads the next character from the keyboard.
     * waits until a key is pressed and then released, then echoe
s
     * the key to the screen, and returns the value of the presse
d key.
     */
    function char readChar() {
```

```
        var char key;
        while( Keyboard.keyPressed() = 0 ) {}
        let key = Keyboard.keyPressed();
        while( ~(Keyboard.keyPressed() = 0) ) {}
        do Output.printChar(key);
        return key;
    }

  function char keyPressed()
  function String readLine(String message)
  function int readInt(String message)
}
```

檔案: Math.jack

```
class Math {
```

```
  function void init()
  function int abs(int x)
  function int multiply(int x, int y)
  function int divide(int x, int y)
  function int min(int x, int y)
  function int max(int x, int y)
  function int sqrt(int x)
}
```

檔案: Sys.jack

```
class Sys {

    /** Performs all the initializations required by the OS. */
    function void init() {
        do Math.init();
```

```
        do Output.init();
        do Screen.init();
        do Keyboard.init();
        do Memory.init();
        do Main.main();
        do Sys.halt();
        return;
    }

    /** Halts execution. */
    function void halt() {
        while(true){}
        return;
    }
function void error(int errorCode)
function void wait(int duration)
```

```
}
```

一旦完成這些作業之後，您就從頭到尾實作出了一台完整的電腦。

現在，在您的腦海中，整台電腦應該是鉅細靡遺，一覽無遺的了。

您有沒有發現，電腦對您而言，開始變得很透明呢？

# 雜誌訊息

## 讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 程式人雜誌社團 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 － 那就是讓讀者根據雜誌的價值捐款給慈善團體。 讀雜誌做公益也不需要有壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！

本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顱顏基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

# 投稿須知

給專欄寫作者： 做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

給網誌捐贈者： 如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

給文章投稿者： 程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每個月 25 日前投稿到程式人雜誌社團 的檔案區，我們會盡可能將稿件編入隔月1號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，我們最希望的格式是採用 markdown 的格式撰寫，然後將所有檔按壓縮為 zip 上傳到社團檔案區給我們， 如您想學習 markdown 的撰寫出版方式，可以參考 [看影片學 markdown 編輯出版流程] 一文。

如果您無法採用 markdown 的方式撰寫，也可以直接給我們您的稿件，像是 MS. Word 的 doc 檔或 LibreOffice 的 odt 檔都可以，我們 會將這些稿件改寫為 markdown 之後編入雜誌當中。

## 參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 程式人雜誌社團 來與我們討論相關事宜。

## 公益資訊

| 公益團體 | 聯絡資訊 | 服務對象 | 捐款帳號 |
| --- | --- | --- | --- |
| 財團法人羅慧夫顱顏基金會 | http://www.nncf.org/<br>lynn@nncf.org<br>02-27190408分機232 | 顱顏患者 (如唇顎裂、小耳症或其他罕見顱顏缺陷） | 銀行：009 彰化銀行民生分行<br>帳號：5234-01-41778-800 |
| 社團法人台灣省兒 | http://www.cyga.org/<br>cyga99@gmail.com | 單親、隔代教養.弱勢及一 | 銀行：新光銀行<br>戶名：台灣省兒童少年成長協會 |

| | | | |
|---|---|---|---|
| 童少年成長協會 | 04-23058005 | 般家庭之兒童青少年 | 帳號：103-0912-10-000212-0 |