

# *Experiment in Compiler Construction*

## *Sinh mã đích(2)*

**Nguyễn Hữu Đức**

Bộ môn Hệ thống thông tin  
Khoa Công nghệ Thông tin  
Đại học Bách khoa Hà nội

# Nội dung

---

- Giới thiệu kplrun
- Giới thiệu instructions.\*, codegen.\*
- Sinh mã (không chương trình con/array)
  - Sinh mã cho lệnh gán
  - Sinh mã cho lệnh if
  - Sinh mã cho lệnh while
  - Sinh mã cho lệnh for
  - Sinh mã cho điều kiện
  - Sinh mã cho biểu thức

# kplrun

---

- Là bộ thông dịch cho máy ngăn xếp

```
$ kplrun <source> [-s=stack-size] [-c=code-size] [-debug] [-dump]
```

- Tùy chọn –s: định nghĩa kích thước stack
- Tùy chọn –c: định nghĩa kích thước tối đa của mã nguồn
- Tùy chọn –dump: In mã ASM
- Tùy chọn –debug: chế độ gõ rối

## kplrun

---

- Tùy chọn –debug: chế độ gõ rối
  - a: địa chỉ tuyệt đối của địa chỉ tương đối (level, offset)
  - v: giá trị tại địa chỉ tương đối (level,offset)
  - t: giá trị đầu ngăn xếp
  - c: thoát khỏi chế độ gõ rối

# Instructions.c

```
enum OpCode {  
    OP_LA,      // Load Address:  
    OP_LV,      // Load Value:  
    OP_LC,      // load Constant  
    OP_LI,      // Load Indirect  
    OP_INT,     // Increment t  
    OP_DCT,     // Decrement t  
    OP_J,       // Jump  
    OP_FJ,      // False Jump  
    OP_HL,      // Halt  
    OP_ST,      // Store  
    OP_CALL,    // Call  
    OP_EP,      // Exit Procedure  
    OP_EF,      // Exit Function  
  
    OP_RC,      // Read Char  
    OP_RI,      // Read Integer  
    OP_WRC,     // Write Char  
    OP_WRI,     // Write Int  
    OP_WLN,     // WriteLN  
    OP_AD,      // Add  
    OP_SB,      // Substract  
    OP_ML,      // Multiple  
    OP_DV,      // Divide  
    OP_NEG,     // Negative  
    OP_CV,      // Copy Top  
    OP_EQ,      // Equal  
    OP_NE,      // Not Equal  
    OP_GT,      // Greater  
    OP_LT,      // Less  
    OP_GE,      // Greater or Equal  
    OP_LE,      // Less or Equal  
  
    OP_BP      // Break point.  
};
```

# Instructions.c

```
struct Instruction_ {  
    enum OpCode op;  
    WORD p;  
    WORD q;  
};
```

```
struct CodeBlock_ {  
    Instruction* code;  
    int codeSize;  
    int maxSize;  
};
```

```
CodeBlock* createCodeBlock(int maxSize);  
void freeCodeBlock(CodeBlock* codeBlock);  
void printInstruction(Instruction* instruction);  
void printCodeBlock(CodeBlock* codeBlock);  
  
void loadCode(CodeBlock* codeBlock, FILE* f);  
void saveCode(CodeBlock* codeBlock, FILE* f);  
  
int emitLA(CodeBlock* codeBlock, WORD p, WORD q);  
int emitLV(CodeBlock* codeBlock, WORD p, WORD q);  
int emitLC(CodeBlock* codeBlock, WORD q);  
....  
int emitLT(CodeBlock* codeBlock);  
int emitGE(CodeBlock* codeBlock);  
int emitLE(CodeBlock* codeBlock);  
  
int emitBP(CodeBlock* codeBlock);
```

# codegen.c

---

```
void initCodeBuffer(void);
void printCodeBuffer(void);
void cleanCodeBuffer(void);
int serialize(char* fileName);

int genLA(int level, int offset);
int genLV(int level, int offset);
int genLC(WORD constant);
...
int genLT(void);
int emitGE(void);
int emitLE(void);
```

# Sinh mã lệnh gán

**v := exp**

```
<code of l-value v>    // đẩy địa chỉ của v lên stack  
<code of exp>          // đẩy giá trị của exp lên stack  
ST
```

# Sinh mã lệnh if

**If <dk> Then statement;**

```
<code of dk>      // đẩy giá trị điều kiện dk lên stack  
FJ L  
<code of statement>  
L:  
...
```

**If <dk> Then st1 Else st2;**

```
<code of dk>      // đẩy giá trị điều kiện dk lên stack  
FJ L1  
<code of st1>  
J L2  
L1:  
<code of st2>  
L2:  
...
```

# Sinh mã lệnh while

**While <dk> Do statement**

```
L1:  
  <code of dk>  
  FJ L2  
  <code of statement>  
  J L1  
L2:  
  ...
```

# Sinh mã lệnh for

**For v := exp1 to exp2 do statement**

```
<code of l-value v>
CV    // nhân đôi địa chỉ của v
<code of exp1>
ST    // lưu giá trị đầu của v
L1:
CV
LI    // lấy giá trị của v
<code of exp2>
LE
FJ L2
<code of statement>
CV;CV;LI;LC 1;AD;ST;    // Tăng v lên 1
J L1
L2:
DCT 1
...
```

## Nhiệm vụ

---

- Đidden vào codegen.c
  - genVariableAddress (Object\* var)  
// Đẩy địa chỉ một biến lên stack
  - genVariableValue (Object\* var)  
// Đẩy giá trị một biến lên stack
- Tạm thời xem các biến đều nằm mức 0  
(trên frame hiện tại)

## Nhiệm vụ

---

- Điền vào parser.c
  - Sinh mã l-value cho biến
  - Sinh mã các câu lệnh: gán, if, while, for
  - Sinh mã điều kiện
  - Sinh mã biểu thức