# 任务

**Вариант N 10**

EX10: PROC OPTIONS ( MAIN );

DCL A DEC FIXED INIT ( 10 );

DCL B BIT ( 16 );

B = A;

END EX10;

将这个程序编译成汇编语言（IBM 370机器）。

# 工作目录及工作文件描述

工作目录：LinuxLabRab.tgz 解压出来的文件

工作目录介绍：

图形用户界面, 应用程序

描述已自动生成

## 1. R\_e\_a\_d\_m\_y

格式：txt

内容：

09.03.2007

1. Данный комплект, состоящий из трех элементов учебной системы программирования

и тестового задания:

- **компилятор** PL/1 --> Ассемблер ( komppl.c ),

- **компилятор** Ассемблер --> объектный образ ( kompassr.c ),

- **абсолютный загрузчик, эмулятор и отладчик** ( absloadm.c ),

- **тестовое задание** ( examppl.pli и spismod ).

2. Для получения исполняемых модулей двух компиляторов и абсолютного загрузчика

следует выполнить Bash-скрипт GenSysProgr.

3. После получения исполняемого кода компиляторов можно запускать на выполнение

тестовый пример с помощью Bash-скрипта StartTestTask.

作用：讲述了完成实验的顺序

## 2. Makefile

内容：

# Script for Programming system generation

**all: komppl.exe kompassr.exe absloadm.exe**

#komppl.o: komppl.c

# gcc -o komppl.o komppl.c

**komppl.exe: komppl.c**

@echo "\_\_\_\_\_\_\_\_k o m p p l . e x e g e n e r a t i o n\_\_\_\_\_\_"

**gcc -o komppl.exe komppl.c**

@echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

**kompassr.exe: kompassr.c**

@echo "\_\_\_\_\_\_\_\_k o m p a s s r . e x e g e n e r a t i o n\_\_\_\_\_\_"

**gcc -o kompassr.exe kompassr.c**

@echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

**absloadm.exe: absloadm.c**

@echo "\_\_\_\_\_\_\_\_a b s l o a d m . e x e g e n e r a t i o n\_\_\_\_\_\_"

**gcc -o absloadm.exe absloadm.c -lncurses**

@echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

使用gcc编译器编译3个c程序：**komppl.c kompassr.c absloadm.c**

## 3. komppl.c

【重点文件】

这个c程序实现了PLI语言的简易编译器。该实验的任务就是完善这个文件，使编译器可以把源代码编译成正确的汇编代码。

## 4. kompassr.c

应该是暂时用不到

## 5. examppl.pli

【重要文件】

输入的源高级语言文件（PL/1语言），

## 6. absloadm.c

应该是暂时用不到

## 7. StartTestTask

脚本，负责执行编译过程。内容：

#! /bin/bash

echo

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_S T A R T J O B\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo

echo ' PL/1 -----> Asswmbler c o m p i l i n g'

echo "continue?(y/n)"

read OTV

if [ "$OTV" = "y" ]

then

echo

**./komppl.exe examppl.pli**

else

echo

exit

fi

echo

echo ' Assembler -----> Object image c o m p i l i n g'

echo "continue?(y/n)"

read OTV

if [ "$OTV" = "y" ]

then

echo

./kompassr.exe examppl.ass

else

echo

exit

fi

echo

echo ' Load, run and debug Object image '

echo "continue?(y/n)"

read OTV

if [ "$OTV" = "y" ]

then

echo

./absloadm.exe spis.mod

else

echo

exit

fi

echo

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_F I N I S H J O B\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

## 8. GenSysProg

脚本，实际上执行的是make

#! /bin/bash

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_S T A R T S Y S T E M G E N E R A T I O N\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

echo

make

echo

echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_F I N I S H S Y S T E M G E N E R A T I O N\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"

## 9. ChangeCodTable

脚本，将文本的ROI-8编码转化成UTF-8编码

#! /bin/bash

iconv -f KOI8-R ./R\_e\_a\_d\_m\_y -o ./R\_e\_a\_d\_m\_y.temp

iconv -f KOI8-R ./komppl.c -o ./komppl.c.temp

iconv -f KOI8-R ./kompassr.c -o ./kompassr.c.temp

iconv -f KOI8-R ./absloadm.c -o ./absloadm.c.temp

rm -f ./R\_e\_a\_d\_m\_y

rm -f ./komppl.c

rm -f ./kompassr.c

rm -f ./absloadm.c

mv ./R\_e\_a\_d\_m\_y.temp ./R\_e\_a\_d\_m\_y

mv ./komppl.c.temp ./komppl.c

mv ./kompassr.c.temp ./kompassr.c

mv ./absloadm.c.temp ./absloadm.c

## 10. spis.mod

未知

examppl.tex

# komppl.c 变量列表和函数列表

## 常量列表

|  |  |
| --- | --- |
| 常量名称 | 常量描述 |
| MAXNISXTXT | 定义源代码最大行数，用于声明源代码存储数组的最大长度 |
| NSINT | 定义语法分析中用到的语法规则的最大数量 |
| NCEL | 定义目标栈的最大深度 |
| NDST | 定义达成目标的栈的最大深度 |
| NVXOD | 定义识别的不同的输入符号的最大数量 |
| NSTROKA | 定义存储压缩后源代码行的字符数组的最大长度 |
| NNETRM | 定义非终结符的最大数量 |
| MAXLTXT | 定义存储汇编语言输出文本行的最大数量 |
| MAXFORMT | 定义用于格式化源代码片段的数组的最大长度 |
| NSYM | 定义符号表中符号的最大数量 |

## 变量列表

|  |  |  |
| --- | --- | --- |
| 变量名称 | 类型 | 描述 |
| NISXTXT | int | 数组长度 |
| ISXTXT | char[MAXNISXTXT][80] | 存储源代码的二维字符数组 |
| I1, I2, I3, I4 | int | 循环计数器 |
| PREDSYM | char | 最后处理的字符 |
| STROKA | char[NSTROKA] | 存储压缩后的源代码 |
| I, J, K, L | int | 当前索引，分别用于**压缩文本、语法规则表、目标栈和目标栈** |
| ASS\_CARD | union | 用于生成输出文件的记录模板 |
| ASSTXT | char[MAXLTXT][80] | 存储汇编语言输出文本的数组 |
| IASSTXT | int | 输出数组的索引 |
| FORMT | char[MAXFORMT][9] | 用于格式化源代码片段的数组 |
| IFORMT | int | 格式化数组的索引 |

## 函数列表及其功能描述

|  |  |
| --- | --- |
| 函数头 | 功能描述 |
| void compress\_ISXTXT() | 压缩源代码文本，移除多余空格 |
| void build\_TPR() | 根据邻接矩阵构建转移矩阵 |
| void mcel(char\* T1, int T2, int T3) | 向目标栈添加新的元素 |
| void mdst(char\* T1, int T2, int T3, int T4, int T5) | 向达成目标的栈添加新的元素 |
| int numb(char\* T1, int T2) | 计算字符串在VXOD表中的索引编号 |
| int sint\_ANAL() | 实现语法分析，构建语法分析树 |
| long int VALUE(char\* s) | 将二进制常量字符串转换成long int类型 |
| void FORM() | 格式化指定的源代码片段 |
| void ZKARD() | 将生成的汇编语言指令添加到输出数组中 |
| int AVI1() | 第一遍语义分析中处理"AVI"（算术表达式） |
| int BUK1() | 第一遍语义分析中处理字母（字符） |
| int CIF1() | 第一遍语义分析中处理数字 |
| int IDE1() | 第一遍语义分析中处理标识符 |
| int IPE1() | 第一遍语义分析中处理变量名 |
| int IPR1() | 第一遍语义分析中处理程序名 |
| int LIT1() | 第一遍语义分析中处理字面量（常量） |
| int MAN1() | 第一遍语义分析中处理浮点数的尾数部分 |
| int ODC1() | 第一遍语义分析中处理PL/1的DCL声明 |
| int OEN1() | 第一遍语义分析中处理PL/1的END结束声明 |
| int OPA1() | 第一 |
| int OPR1() | 第一遍语义分析中处理PL/1的PROC过程声明 |
| int PRO1() | 第一遍语义分析中处理整个PL/1程序 |
| int RZR1() | 第一遍语义分析中处理变量的精度声明 |
| int TEL1() | 第一遍语义分析中处理PL/1程序的主体部分 |
| int ZNK1() | 第一遍语义分析中处理算术操作符 |
|  |  |

# 实验步骤

1. 修改词法分析器的数据库

2. 修改语义分析器的数据库

2.1 修改SINT [ NSINT ]

2.2 修改VXOD [ NVXOD ]

2.3 修改char TPR [ NVXOD ] [ NNETRM ]

2.4 修改函数：: DCF1, DCF2, ODC1, AVI2, OEN2, OPA2, OPR2, RZR2

2.4.1 **DCF1**

2.4.2 **ODC1**

2.4.3 **AVI2**

2.4.4 **OEN2**

2.4.5 **OPA2**

2.4.6 **OPR2**

2.4.7 **RZR2**

2.4.8 **DCF2**

3. 运行程序，获取结果

# 编译原理（IBM 360/370）

应用程序的命令、数据和工作数据在计算机RAM中的放置（Размещение команд, данных и рабочих данных прикладной программы в ОЗУ ЭВМ）

**（ALL LECTURES P23）**

文本

描述已自动生成

表格

描述已自动生成

表格

描述已自动生成

文本

描述已自动生成

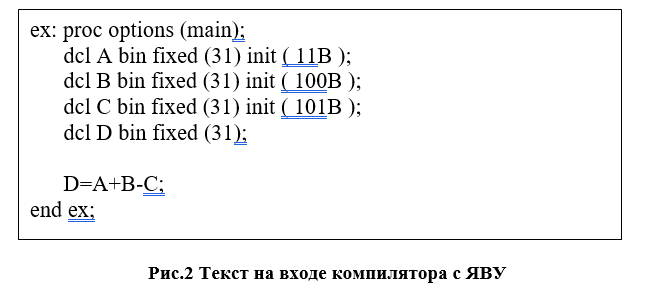
表格

描述已自动生成

表格

低可信度描述已自动生成

# 《Recommends》中的例子



**Синтаксические правки**

Синтаксис демо-примера (см. стр. 7 и стр.8 Пособия) имеет следующее определение (см. рис. 6):

1. <PRO> ::= <OPR><TEL><OEN>

2. <OPR> ::= <IPR>:PROC\_OPTIONS(MAIN);

3. <IPR> ::= <IDE>

4. <IDE> ::= <BUK> | <IDE><BUK> | <IDE><CIF>

5. <BUK> ::= A | B | C | D | E | M | P | X

6. <CIF> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

7. <TEL> ::= <ODC> | <TEL><ODC> | <TEL><OPA>

8. <ODC> ::= DCL\_<IPE>\_BIN\_FIXED(<RZR>); | DCL\_<IPE>\_BIN\_FIXED(<RZR>)INIT(<LIT>);

9. <IPE> ::= <IDE>

10. <RZR> ::= <CIF> | <RZR><CIF>

11. <LIT> ::= <MAN>B

12. <MAN> ::= 1 | <MAN>0 | <MAN>1

13. <OPA> ::= <IPE>=<AVI>;

14. <AVI> ::= <LIT> | <IPE> | <AVI><ZNK><LIT> | <AVI><ZNK><IPE>

15. <ZNK> ::= + | -

16. <OEN> ::= END\_<IPR>

Здесь использованы следующие метасимволы и символы:

"<" и ">" - левый и правый ограничители нетерминального символа,

"::=" - метасимвол со смыслом "равно по определению",

"|" - метасимвол альтернативного определения "или",

"\_" - терминальный символ со смыслом "пробел",

"<PRO>" - нетерминал "программа",

"<OPR>" - нетерминал "оператор пролога программы",

"<IPR>" - нетерминал "имя программы",

"<IDE>" - нетерминал "идентификатор",

"<BUK>" - нетерминал "буква",

"<CIF>" - нетерминал "цифра",

"<TEL>" - нетерминал "тело программы",

"<ODC>" - нетерминал "оператор declare",

"<IPE>" - нетерминал "имя переменной",

"<RZR>" - нетерминал "разрядность",

"<LIT>" - нетерминал "литерал",

"<MAN>" - нетерминал "мантисса",

"<OPA>" - нетерминал "оператор присваивания арифметический",

"<AVI>" - нетерминал "арифметическое выражение",

"<ZNK>" - нетерминал "знак",

"<OEN>" - нетерминал "оператор эпилога программы".

转化成箭头图：

1. <OPR><TEL><OEN> ──> <PRO>

2. <IPR>:PROC\_OPTIONS(MAIN); ──> <OPR>

3. <IDE> ──> <IPR>

4. <BUK> | <IDE><BUK> | <IDE><CIF> ──> <IDE>

5. A | B | C | D | E | M | P | X ──> <BUK>

6. 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ──> <CIF>

7. <ODC> | <TEL><ODC> | <TEL><OPA> ──> <TEL>

8. DCL\_<IPE>\_BIN\_FIXED(<RZR>); | DCL\_<IPE>\_BIN\_FIXED(<RZR>)INIT(<LIT>); ──> <ODC>

9. <IDE> ──> <IPE>

10. <CIF> | <RZR><CIF> ──> <RZR>

11. <MAN>B ──> <LIT>

12. 1 | <MAN>0 | <MAN>1 ──> <MAN>

13. <IPE>=<AVI>; ──> <OPA>

14. <LIT> | <IPE> | <AVI><ZNK><LIT> | <AVI><ZNK><IPE> ──> <AVI>

15. + | - ──> <ZNK>

16. END\_<IPR> ──> <OEN>

拆成左部为单个表达式的形式：

<OPR><TEL><OEN> ──> <PRO>

<IPR>:PROC\_OPTIONS(MAIN); ──> <OPR>

<IDE> ──> <IPR>

<BUK> ──> <IDE>

<IDE><BUK> ──> <IDE>

<IDE><CIF> ──> <IDE>

A ──> <BUK>

B ──> <BUK>

C ──> <BUK>

D ──> <BUK>

E ──> <BUK>

M ──> <BUK>

P ──> <BUK>

X ──> <BUK>

0 ──> <CIF>

1 ──> <CIF>

2 ──> <CIF>

3 ──> <CIF>

4 ──> <CIF>

5 ──> <CIF>

6 ──> <CIF>

7 ──> <CIF>

8 ──> <CIF>

9 ──> <CIF>

<ODC> ──> <TEL>

<TEL><ODC> ──> <TEL>

<TEL><OPA> ──> <TEL>

DCL\_<IPE>\_BIN\_FIXED(<RZR>); ──> <ODC>

DCL\_<IPE>\_BIN\_FIXED(<RZR>)INIT(<LIT>); ──> <ODC>

<IDE> ──> <IPE>

<CIF> ──> <RZR>

<RZR><CIF> ──> <RZR>

<MAN>B ──> <LIT>

1 ──> <MAN>

<MAN>0 ──> <MAN>

<MAN>1 ──> <MAN>

<IPE>=<AVI>; ──> <OPA>

<LIT> ──> <AVI>

<IPE> ──> <AVI>

<AVI><ZNK><LIT> ──> <AVI>

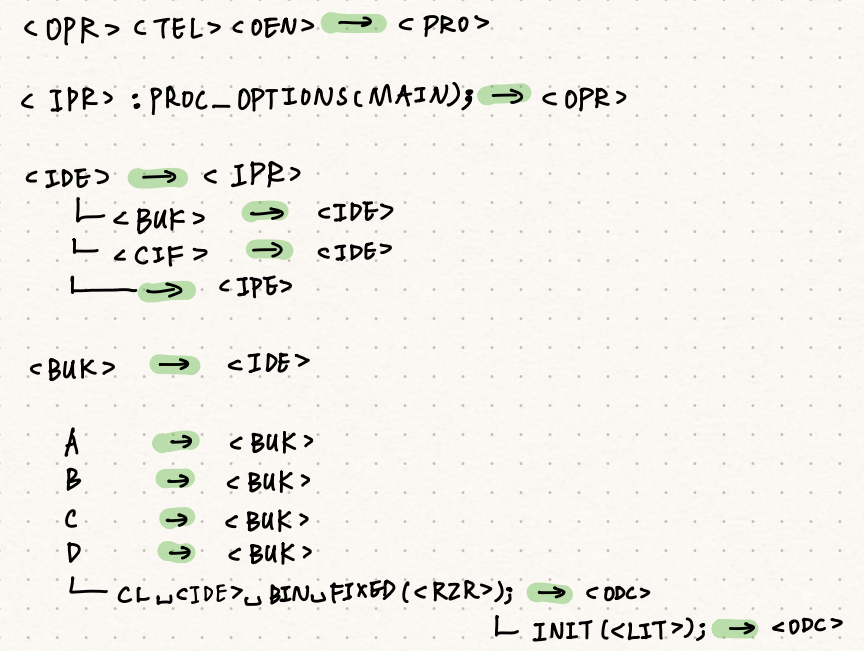
<AVI><ZNK><IPE> ──> <AVI>

+ ──> <ZNK>

- ──> <ZNK>

END\_<IPR> ──> <OEN>

合并共同的左部：



文本

中度可信度描述已自动生成

文本, 信件

描述已自动生成

信件

低可信度描述已自动生成

文本, 信件

描述已自动生成

Table of Entrance （按照同学文档中）

0

1

2

3

4

5

6

7

8

9

A

B

C

D

DCL A BIN FIXED (31);

DCL A BIN FIXED (31) INIT (10B);

DCL A CHAR (31);

DCL

E

M

P

X

BUK

IDE

+

-

IPR

CIF [修改入口]

RZR

MAN

IPE

LIT

AVI

OPR

ODC

TEL

CHAR [..?]

DEC FIXED INIT(DCF);ODC

BIT(RZR);ODC

DCF

DCF

### 入口表：

**DCF 234 (WHY 234 BUT NOT 230?!)**

**BUK 93**

**IDE 96**

**IPR 113**

**CIF 136**

**RZR 139**

**MAN 143**

**IPE 153**

**LIT 161**

**AVI 164**

**OPR 171**

**ODC 176**

**TEL 179**

**ZNK 0 (WHY?)**

**OEN 0 (WHY?)**

**OPA 0 (WHY?)**

**OPR 0 (WHY?)**

A 33

B 36

C 39

**D 42**

E 74

M 84

P 87

X 90

0 1

1 4

2 9

3 12

4 15

5 18

6 21

7 24

8 27

9 30

+ 107

- 110

:

I

R

N

O

T

S

(

)

\_

;

L

F

=

H

**\* 198**