НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Кафедра системного програмування і спеціалізованих комп'ютерних систем

КУРСОВА РОБОТА

з дисципліни «Системне програмування»
на тему: Розробка компілятора програм мовою Асемблера

Студента (ки) _II_ курсу _КВ-84__ групи
за спеціальностю
123 «Комп'ютерна інженерія»
Голуб В. В.
(прізвище та ініціали)
Керівник : кандидат технічних наук, старший науковий співробітник доцент кафедри спеціалізованих комп'ютерних систем
Тесленко О.К.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)
_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)
_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ- 2020 рік

2.Індивідуальне завдання

Варіант 11.

Ідентифікатори

Містять великі і малі букви латинского алфавіту та цифри. Починаються з букви. Великі та малі букви не відрізняються.

Константи

Шістнадцятерічні, десяткові та
двійкові константи

Директиви

END,
SEGMENT - без операндів, ENDS, программа може мати тільки один сегмент кодів і тільки
один сегмент даних
DB,DW,DD з одним операндом - довільний арифметичний вираз над константами

Розрядність даних та адрес

16 - розрядні дані та зміщення в сегменті, 32 -розрядні дані та зміщення не використовуються

Адресація операндів пам'яті

Індексна адресація (Val1[si],Val1[bx] і т.п.)

Заміна сегментів

Префікси заміни сегментів можуть задаватись явно.

Машинні команди

Stosb
Push **reg**
Mul **mem**
Add **reg,mem**
And **reg,reg**
Cmp **mem,reg**
Sub **reg,imm**
Test **mem,imm**
Jne

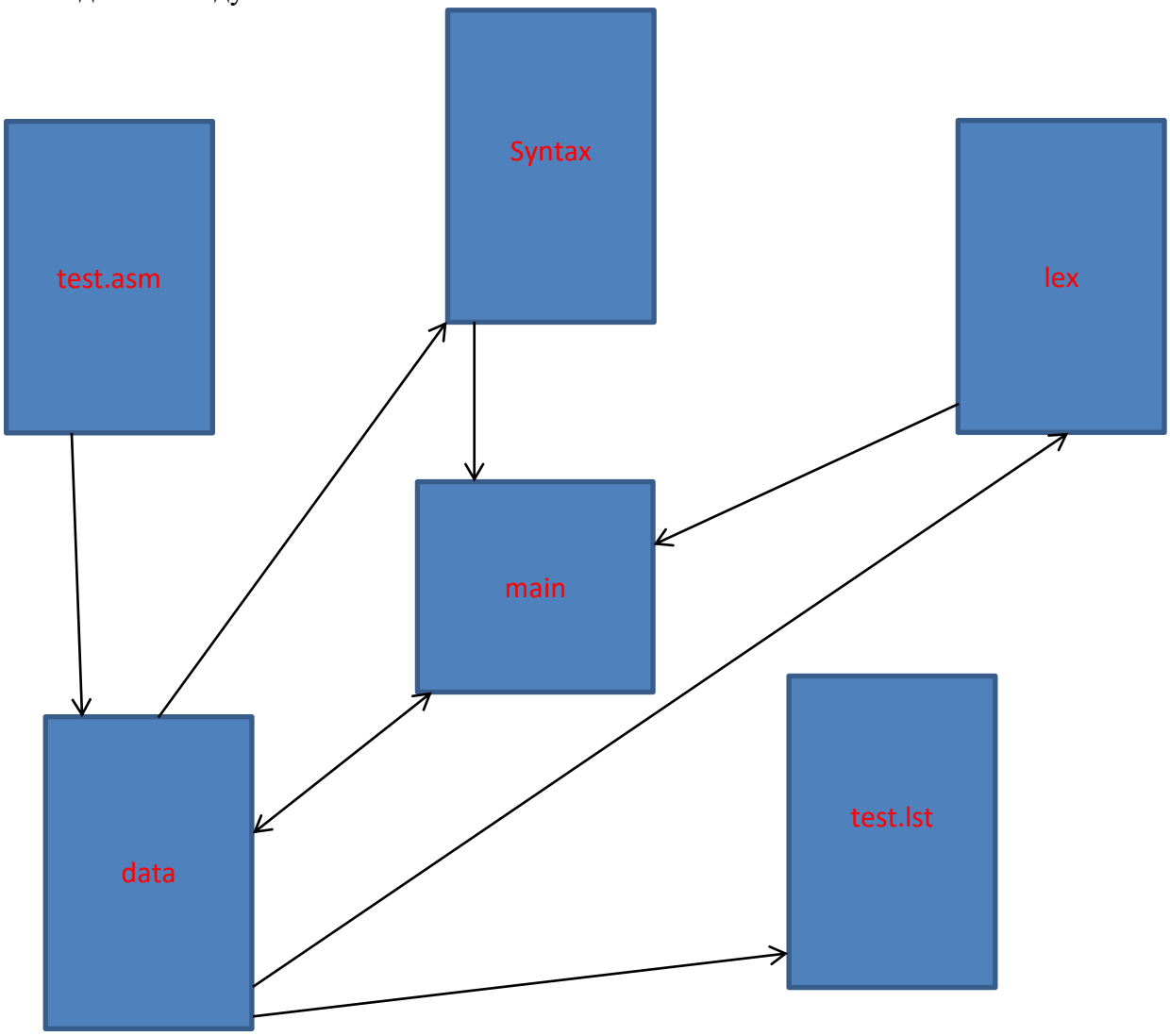
Де **reg –** 8 або 16-розрядні РЗП
mem – адреса операнда в пам’яті
imm - 8 або 16-розрядні безпосередні дані (довільний арифметичний вираз над константами)

3.Опис загальної структури розробленої програми, окремих модулів і підпрограм та їх взаємодії.

Програма складається з 4 модулів та 2 файли один із них test.asm (вхідний файл) та test.lst (вихідний файл, результат роботи програми).

Програма написана методом функціонального програмування , тобто вся реалізація програми базується на функціях.

Взаємодія між модулями



Мета програми: провести аналіз лексичний , структурний вхідного файлу програми на мові асемблера та вивести файл лістингу що відобразить наявність або відсутність помилок в тестовому файлі. Кількість виділених байт , коди команд, зміщення , таблиці ідентифікаторів та сегментів .

Модулі :

1. **main.py** міст лише 1 головну функції main() що починає запуск всієї програми. У цій функції відбувається відкриття вхідного файлу test.asm та створення файлу для запису вихідного файлу програми.
   До даного модуля підключенні інші модулів із яких у функції main() викликаються допоміжні функції для аналізу , виводу , обробки вхідного файлу.

2. **data.py** модуль який слугує місцем зберігання об'єктів для аналізу вхідного файлу.
   WAYTOASM(WAYTOLST) – шлях до вхідного і вихідного файлу.
   Lexemes, Memmory – данні для аналізу команд ,символів, що належать алфавіту асмеблера , визначення директив, визначення типу пам'яті та типу змінних які використовує модуль **lex.py , main.py, syntax.py** . для аналізу вхідного файлу. Також містить допоміжні змінні для інших функції що використовують інші модулі програми.

3. **lex.py** Молуль що виконує лексичний аналіз вхідного файлу. Містить функції :
   ParseFileToList(line) – Розбиває кожен рядок вхідного файлу виключаючи із нього пробіли, табуляцію , елементи що знаходиться між пробілами (таблуяцією) стають елементами списку що в результаті повертає функція.
   SetLexemeType(lexeme) – проводить аналіз списку що повертає функція **ParseFileToList(line)** і визначає тип лексем та повертає список із характеристикою кожної лексеми.
   DelFromList(lst, start, count)- видаляє елементи із списку lst що розміщуються до номера count включаючи його.

4. **syntax.py** – модуль що місититиь в собі основну частину функцій які виконують аналіз лексичний, елементи першого та другого переглядів
   Syntax(senline)- проводить синтаксичний аналіз аргументу senline і повертає результат аналізу.

   operat(oper) – за данним аргументу визначає що це аргумень чи операції + - / * або круглі дужки. використовується у алгоритмі Дейкстри для автоматичного обчислення виразів.
   dexstra(math) - функція реалізовує алгоритм дейксти дл яобчислення математичних виразів, функція повертає результат обчислення.

   Segmen_16reg_or_8seg(regist) – проводить аналіз переданого регістру 16 ібо8 розрядного та повертає його код який використовується для формування байту MOD R/M , у разі якщо подано назву регістру функція повертає повідомленя 'error'

   M_R(RM) – функція що повертає код який використовуєтсья при адресації типу var[bx] а саме аналізує [bx] і при наявності відповдності повертає їхній код який є частиною байту mod r/m і саме r/m у разі відсутності відповідності повертається повідомлення про помилку 'error'

PREfSEG(pref)- функція аналізує аргумент префікцсу заміни сегментів і у разі виявлення співпадіння повертає код цього регістру згідно мови асемблеру, у разі відсутності повертає повідомлення про помилку ‘error’

sen_offset(senline, struc_senline,under,line)- функція формує байти зміщення команд в сегменті, записує у таблицю ідентифікаторів данні. Приймає попереднє зміщення в сегменті under, senline –список з яких елементів складаєтсья один рядок що аналізуємо , sstruc_senline- структура данного рядка, line – список в якому описано тип елементв які записані в senline.функція повертає результат зміщення в сегменті для поточного рядка.

STOSB(),PUSH(senline), MUL(senline),AND(senline),CMP(senline),MOV(senline) SUB(senline),TEST(senline),JNE(offset, lable_name) функції що викликаються функцією **sen_offset(senline, struc_senline,under,line)** якщо у senline зустрічаються команди що відповідають назвам цих функцій. Які у свою чергу повертають розмір що займає команда,(назви функцій співпадають із функціями)

PRINT_TABL_IDENT() –функція що повертає відформатовану таблицю ідентифікаторів.

PRINT_LST() – функція що повертає результат роботи усієї програми після повного аналізу вхідного файлу, що у свою чергу записується до файлу вихідного.

find_err(var) –функція що повертає код помилки якщо не знайдено лексему END передчасно (тобто після неї розміщуються ще рядки програми на мові асемблера) якщо все добре повертається 0 якщо помилка 1.

Second(senline, struc_senline,under,line) – функція що реалізує 2 перегляд, тобто повертає поточний рядок вхідного файлу , у якому уже визначено код комнади , зміщення всегменті, байт mod r/m , зміщення зміних які використовуютсья у командах, значення які мають змінні при оголошенні.

write_str(senline) – функція що повертає запис списку senline. Використовується функцією **Second(senline, struc_senline,under,line)** другий перегляд для запису поточного рядка після визначення всіх даних що потребує 2 перегляд (вказано в описі функції другого перегляду перегляду.

find_ID(find) – пошук у таблиці ідентифікаторів аргумента find у разі знаходження повертається ідекс його положення від початку таблиці, якщо не знайдено повертається повідомлення ‘error’

Normal_var_write(var) –виконує нормалізований запис значення числа яке використовується у команді, або при ініціалізації змінної її початкове значення.Повертає вже відредаговений запис числа.

STOSB_2(senline), PUSH_2(senline,line),MUL_2(senline,line, struc_senline) – функції що роблять аналіз команд STOSB , PUSH, MUL якщо вони зумтрічаються у вхідному файлі. Повертають повний повний аналіз рядка каманди. Код команди , значення байту mod r/m ,зміщення змінної в сегменті де вона ініціалізована що використовується як операнд в команді .

Pr_colum_2(colum_2) - функція що переводить двійковий код команди у шістнадцятковий код , якщо переданий аргумент не містить вираз ‘error’ ,в іншому випадку повертає повідомлення про помилку ‘error’

COL_2(f,f1, senline, line) -робить те що і функція **Pr_colum_2(colum_2)** тільки для функції **SUB_2 (…).**

MOD_R_M(f1, f2,senline, line) – формує і повертає двійкове значення байту mod r/m для поточного рядку, у разі помилки при формуванні повертає повідомлення про помилку ‘error’

find_MOD_R_M(f,senline,line)- робить аналіз операнда і повертає що він із себе представляє у байті mod r/m тобто чи він reg, r/m або imm, вразі не знаходження підходящого варінту повертає повідомлення про помилку ‘error’

des(f, senline, line) – повертає значення mod (частина байту mod r/m) тобто проводить аналіз рядка визначає режим адресації а повертає її код у двійковій системі числення.

other(var) – повертає зміщення змінної в сегменті де вона об’явлена, при умові що ця змінна існує і використовується в команді.

other_2(var) – повертає значення числа яке стоїть в команді в якості операнда.

check_vars(var)- перевіряє тип числа, якщо присутня помилка повертає повідомлення про помилку ‘error’

ADD_2(senline,line, struc_senline), AND_2(senline,line, struc_senline), CMP_2(senline,line, struc_senline), MOV_2(senline,line, struc_senline), TEST_2(senline,line, struc_senline), SUB_2(senline,line, struc_senline)- фукнції викликаються у функції Second(…) якщо поточний досліджуваний рядок містить команди ADD , AND, CMP, MOV, TEST, SUB , та повертають готовий рядок із кодом команди, байтом mod r/m при використанні зміної її зміщення в сегменті де вони об’явлені та ініціалізовані., або занчення числа який використовуєтсья в якось операнда.

JNE_2(senline,line, struc_senline) – при знаходженні в поточному досліджуваному рядку команди JNE , викликається дана функція в функції Second(...) та повертає рядок із записом про код команди ,байт mod r/m ., вразі не знаходження операнда , повертається повідомлення про помилку ‘error’.

JNE_90(var) – функція перевіряє чи не виділено зайві байти для мітки якщо так , записується комнда 90 90 що заповнює ці байти.

comp_comand(cmnd,first_operand,second_operand) – функція що аналізує cmnd назва команди, та коди опернідів першого та другого повертає код команди для відповідної назви команди, і типу її операндів, або повідомлення про помилку.

oper_1(f,command,senline,line,struc_senline) – проводить аналіз операнду в залежності чи це регістр, чи змінна, чи вираз, або число, та повертає відповідний код.

Pref_seg (pref) -аналізує чи аргумент що переданий у функцію є префіксом заміни сегментів, або повертає його, або повертає повідомлення про помилку.

MEM_Chack(number,type) – перевіряє правильність ініціалізаціїї змінних ,відповідно до їх значення ,та типу змінної в яку ми це значення записуємо ,уразі помилки повертається повідомлення про помилку ‘error’ або значення True.

help(helpp) – допоміжна функція що перевіряє чи цифра належить до десяткової системи числення.

write_VAR(number,line)-функція що визначає тип запису числа, для і повертає нормалізоване значення цього числа, або помилку.

convert_base(num, to_base=10, from_base=16)- конвертує число із 10-ї системи в 16-ву систему числення.

DecToHex(off) –нормалізує вигляд числа та перетворює число із 10системи числення в 16систему числення .

4. Результати кожного з етапів курсової роботи

І етап (тестовий файл)

```
DATA segment
        Var db 0fch
        Var2 dw 0340h
        Var3 dd 0a7f5ch
        Var4 db 01b
        Var5 db 5
DATA ends
CODE Segment
label1:
        stosb
        push ax

        mul Var[si]

        add ax, Var2[di]

        and ax, bx

        cmp Var2[bx], bx

        Jne label2

label2:

        mov ax , ES:Var2[si]

    sub al, ( 124 + 23 * ( 98 / 45 ) – 62 )

        test Var4[bx], 01b

        Jne label2

CODE ends
END
```

II етап

Вивід лексера із структурою речення
l – мітка
N- ім’я
M-мнемокод
O-операнд

DATA segment
N:1 M:2 1

Var db 0fch
N:1 M:2 1 O:3 1

Var2 dw 0340h
N:1 M:2 1 O:3 1

Var3 dd 0a7f5ch
N:1 M:2 1 O:3 1

Var4 db 01b
N:1 M:2 1 O:3 1

Var5 db 5
N:1 M:2 1 O:3 1

DATA ends
N:1 M:2 1

CODE Segment
N:1 M:2 1

label1 :
l:1

stosb
M:1 1

push ax
M:1 1 O:2 1

mul Var[si]
M:1 1 O:2 4

add ax, Var2[di]
M:1 1 O:2 1 O:4 4

and ax, bx
M:1 1 O:2 1 O:4 1

cmp Var2[bx], bx
M:1 1 O:2 4 O:7 1

Jne label2
M:1 1 O:2 1

label2:
l:1

mov ax , ES:Var2[si]
M:1 1 O:2 1 O:4 6

sub al, (124+23*(98/45)-62)
M:1 1 O:2 1 O:4 13

test Var4[bx], 01b
M:1 1 O:2 4 O:7 1

Jne label2
M:1 1 O:2 1

CODE ends
N:1 M:2 1

END

M:1 1

III етап (I перший перегляд)

```
0000          DATA    SEGMENT
0000          VAR DB 0FCH
0001          VAR2 DW 0340H
0003          VAR3 DD 0A7F5CH
0007          VAR4 DB 01B
0008          VAR5 DB 5
0009          DATA ENDS
0000          CODE    SEGMENT
0000          LABEL1 :
0000          STOSB
0001          PUSH AX
0002          MUL VAR [ SI ]
0006          ADD AX , VAR2 [ DI ]
000A           AND AX , BX
000C           CMP VAR2 [ BX ] , BX
0010          JNE LABEL2
0014          LABEL2 :
0014          MOV AX , ES : VAR2 [ SI ]
0019          SUB AL , ( 124 + 23 * ( 98 / 45 ) - 62 )
001B           TEST VAR4 [ BX ] , 01B
0020          JNE LABEL2
0022          CODE ENDS
            END
```

NAME	TYPE	VALUE	ATTR
VAR....	DB	0000	DATA
VAR2....	DW	0001	DATA
VAR3....	DD	0003	DATA
VAR4....	DB	0007	DATA
VAR5....	DB	0008	DATA
LABEL1..	NEAR	0000	CODE
LABEL2..	NEAR	0014	CODE

NAME	SIZE	LENGTH	COMBINE CLASS
DATA...	16 bit	0009	PARA NONE
CODE...	16 bit	0022	PARA NONE

IV етап (II перегляд)

Файл другого перегляду моеї програми (без помилок)

```
0000                DATA   SEGMENT
 0000   FC           VAR   DB   0FCH
 0001   0340          VAR2   DW   0340H
 0003   000A7F5C          VAR3   DD   0A7F5CH
 0007   01          VAR4   DB   01B
 0008   05          VAR5   DB   5
 0009               DATA   ENDS
 0000               CODE   SEGMENT
 0000               LABEL1   :
 0000   AA          STOSB
 0001   50         PUSH   AX
 0002   F6  A4 0000   MUL   VAR   [   SI   ]
 0006   03  85 0001       ADD   AX   ,   VAR2   [   DI   ]
 000A    23  C3           AND   AX   ,   BX
 000C    39  9F  0000         CMP   VAR   [   BX   ]   ,   BX
 0010    75  02  90 90       JNE   LABEL2
 0014               LABEL2   :
 0014    26: 8B  84 0001         MOV   AX   ,   ES   :   VAR2   [   SI   ]
 0019    2C  6C           SUB   AL   ,   (   124   +   23   *   (   98   /   45   )   -   62   )
 001B    F6  87 0000      01 TEST   VAR   [   BX   ]   ,   01B
 0020    75 F2               JNE   LABEL2
 0022               CODE   ENDS
              END
```

NAME	TYPE	VALUE	ATTR
VAR....	DB	0000	DATA
VAR2....	DW	0001	DATA
VAR3....	DD	0003	DATA
VAR4....	DB	0007	DATA
VAR5....	DB	0008	DATA
LABEL1..	NEAR	0000	CODE
LABEL2..	NEAR	0014	CODE

NAME	SIZE	LENGTH	COMBINE CLASS
DATA...	16 bit	0009	PARA NONE
CODE...	16 bit	0022	PARA NONE

лістинг якщо є помилки

```
 0000                DATA   SEGMENT
0000    error           VAR   DB   0FLH

     ERROR
0001    0340            VAR2   DW   0340H
0003    000A7F5C            VAR3   DD   0A7F5CH
0007    01            VAR4   DB   01B
0008    05            VAR5   DB   5
0009                DATA   ENDS
0000                CODE   SEGMENT
0000                LABEL1   :
0000    AA          STOSB
0001    50          PUSH   AX
0002    F6  A4 0000   MUL   VAR   [   SI   ]
0006    F7    0000  MUL   VAR1   [   SI   ]   error

     ERROR
000A    03  85 0001        ADD   AX   ,   VAR2   [   DI   ]
000E    23  C3            AND   AX   ,   BX
0010    error  C3           AND   AX   ,   BL

     ERROR
0012    39  9F  0000          CMP   VAR   [   BX   ]   ,   BX
0016    75  02  90 90        JNE   LABEL2
001A                LABEL2   :
001A    26: 8B  84 0001          MOV   AX   ,   ES   :   VAR2   [   SI   ]
001F    8B  error           MOV   AX   ,   KS   :   VAR2   [   SI   ]

     ERROR
0023    2C  6C            SUB   AL   ,   (   124   +   23   *   (   98   /   45   )   -   62   )
0025    F6  error 0000      01 TEST   VAR   [   BX   ,   01B

     ERROR
002A    75          JNE   LABEL   error

     ERROR
002E                CODE   ENDS
                END

NAME                    TYPE            VALUE           ATTR
 VAR....                DB                      0000            DATA
VAR2....                DW                      0001            DATA
VAR3....                DD                      0003            DATA
VAR4....                DB                      0007            DATA
VAR5....                DB                      0008            DATA
LABEL1..                NEAR            0000            CODE
LABEL2..                NEAR            001A            CODE

NAME                    SIZE            LENGTH                  COMBINE CLASS
DATA...                         16 bit          0009            PARA    NONE
CODE...                         16 bit          002E            PARA    NONE
```

5. Роздруківка лістингу модифікованого тесту. (результат трансляції модифікованого тесту TASM).

```
Turbo Assembler        Version 3.1        04/26/20 22:37:05      Page 1
test.ASM

 1                                  .386
 2
 3      0000                        DATA segment use16
 4      0000  FC                          Var db          0fch
 5      0001  0340                        Var2 dw 0340h
 6      0003  000A7F5C                    Var3 dd 0a7f5ch
 7      0007  01                          Var4 db 01b
 8      0008  05                          Var5 db 5
 9      0009                        DATA ends
10                                                  Assume          Cs:Code, Ds:Data
11      0000                        CODE Segment use16
12      0000                        label1:
13      0000  AA                          stosb
14      0001  50                          push ax
15
16      0002  F6 A4          0000r                 mul Var[si]
17
18      0006  03 85 0001r                   add ax, Var2[di]
19
20      000A  23 C3                         and ax, bx
21
22      000C  39 9F          0001r          cmp Var2[bx], bx
23
24      0010  75 02 90 90                   Jne label2
25
26      0014                        label2:
27
28      0014  26: 8B 84 0001r               mov ax          ,ES:Var2[si]
29
30      0019  2C 6C                           sub al, ( 124 + 23  * ( 98 / 45 ) -  62 )
31
32      001B  F6 87          0007r 01       test Var4[bx],01b
33
34      0020  75 F2                         Jne label2
35
36      0022                        CODE ends
37
38                                  END


Turbo Assembler        Version 3.1        04/26/20 22:37:05      Page 2
Symbol Table



Symbol Name                     Type   Value
```

```
??DATE                          Text   "04/26/20"
??FILENAME                      Text   "test     "
??TIME                          Text   "22:37:05"
??VERSION                   Number 030A
@CPU                        Text   0F0FH
@CURSEG                         Text   CODE
@FILENAME                       Text   TEST
@WORDSIZE                       Text   2
LABEL1                          Near   CODE:0000
LABEL2                          Near   CODE:0014
VAR                         Byte   DATA:0000
VAR2                        Word   DATA:0001
VAR3                        Dword          DATA:0003
VAR4                        Byte   DATA:0007
VAR5                        Byte   DATA:0008

Groups & Segments           Bit Size Align  Combine Class

CODE                        16  0022 Para           none
DATA                        16  0009 Para           none
```