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 2 курсу КВ-92 групи

за спеціальністю

123 «Комп’ютерна інженерія»

Оридорога М. О

Керівник старший викладач

Дробязко І. П.

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- 2021 рік

Варіант індивідуального завдання - 16

Ідентифікатори

Містять великі букви латинского алфавіту та цифри. Починаються з букви. Довжина ідентифікаторів не більше 6 символів

Константи

Шістнадцяткові, десяткові та текстові константи

Директиви

END,

SEGMENT - без операндів, ENDS, PROC, ENDP

DB, DW, DD з одним операндом - константою (текстові константи тільки для DB)

Директиви DW та DD в якості операнда можуть мати ім'я або мітку

Розрядність даних та адрес

16- розрядні дані та зміщення в сегменті, у випадку 32 -розрядні даних та зміщень генеруються відповідні префікси зміни розрядності

Адресація операндів пам'яті

*Пряма адресація та базова індексна адресація із множником і з константним зміщенням ([edx+esi*4+6],[ebx+ecx*2+12] і т.п.) з оператором визначення типу (ptr) при необхідності*

Заміна сегментів

Префікси заміни сегментів можуть задаватись явно, а при необхідності автоматично генеруються транслятором

Машинні команди

Ret

Push ***mem***

Pop ***reg***

Or ***reg, reg***

Imul ***reg, mem, imm***

Mov ***reg,mem***

Mov ***mem,imm***

Jz

Call (з прямою адресацією, внутрішньо-сегментні та міжсегментні)

Де ***reg –*** *8, 16 або 32-розрядні РЗП*

mem – *адреса операнда в пам'яті*

imm – *8, 16 або 32-розрядні безпосередні дані (константи)*

Структура програми та загальний опис окремих методів

Файл **main.cpp** – точка входу в програму

1. Етап лексичного аналізу

На даному етапі аналізуються всі вхідні дані, проводиться повний парсинг лексем, визначення їх типу, та визначення загальної структури речення, тобто визначається чи було присутнє поле мнемокоду, наявність операндів та міток.
Модуль **lexer** розбиває кожну строку файла на окремі лексеми.
Модуль **dictionary** виступає в ролі бази даних, а також містить функціїї, які визначають тип конкретної лексеми.
Модуль **translator** виступає в ролі загального модуля, звідси викликаються всі попередні модулі, використовується інформація, яка надається після обробки речення вищезгаданими частинами програми, і базуючись на цих даних, формуються таблиці лексичного аналізу

2. Етап першого проходу

На даному етапі виконується аналіз кожного речення, базуючись на попредньому етапі, і формується зміщення для кожної команди, тобто рахується, скільки місця повинна зайняти в пам’яті та чи інша команда.
Головним у програмі є клас translator. Саме він генерує лістинг методами CreateListing. Спочатку він парсить текст на лексеми, які потім аналізує. Спочатку аналізує через функції, які перевіряють на кількість лексем, першу лексему рядка та чи є ця лексема інструкцією, правильність адреси, порядок аргументів, існування певної команди, наявність лейблу в сегменті, переповненення змінної певного розміру. Під час аналізу генерується офет команди. Назви функцій співпадають з командами які вони перевіряють. Також для зручності були використані абстракції Variable, Segment.

Тестування програми

-Варіант лістингу, сформованого даною програмою

```
Assembly Translator.
Written by Orydoroha Mykhailo.

Filename: test.asm


     1     0000                       DATA SEGMENT
     2     0000                       STRING DB "LOREM"
     3     0005                       VAR0BD DB 123D
     4     0006                       VAR0WH DW 0AB12H
     5     0008                       VAR0DH DD 06E6CCH
     6     000C                       DATA ENDS
     7     0000                       CODE1 SEGMENT
     8     0000                       PROC1 PROC FAR
     9     0000                       ;  IMUL TESTS
    10     0000                       IMUL AX , WORD PTR [ EDX + ESI * 4 +
6 ] , 10
    11     0006                       IMUL EAX , DWORD PTR [ EDX + ESI * 4 +
6 ] , 10
    12     000D                       IMUL AX , WORD PTR [ EDX + ESI * 4 +
6 ] , 1000
    13     0014                       IMUL EAX , DWORD PTR [ EDX + ESI * 4 +
6 ] , 33000
    14     001E                       ;  PUSH TESTS
    15     001E                       PUSH DWORD PTR [ EDX + ESI * 4 + 6 ]
    16     0024                       PUSH WORD PTR [ EDX + ESI * 4 + 6 ]
    17     0029                       ;  MOV TESTS
    18     0029                       MOV BYTE PTR [ EDX + ESI * 4 + 6 ] , 10
    19     002F                       MOV EAX , [ EDX + ESI * 4 + 6 ]
    20     0035                       POP EAX
    21     0037                       RET
    22     0038                       PROC1 ENDP
    23     0038                       ;
    24     0038                       ;  OR TESTS
    25     0038                       OR AX , BX
    26     003A                       OR EAX , EBX
    27     003D                       OR AL , BL
    28     003F                       OR BX , AX ;  VALID
    29     0041                       MOV SI , [ EDX + ESI * 4 + 6 ]
    30     0046                       JZ VOL
    31     004A                       VOL :
    32     004A                       CODE1 ENDS
    33     0000                       CODE2 SEGMENT
    34     0000                       PROC2 PROC
    35     0000                       RET
    36     0001                       PROC2 ENDP
```

```
37      0001                    OR CL , AH
38      0003                    CALL PROC2
39      0006                    CALL PROC1
40      000B                    CODE2 ENDS
41      000B                    END
```

-Варіант сформований за допомогою tasm

Turbo Assembler Version 2.5 06/03/21 17:08:50 Page 1
TEST1.ASM

```
 1                              .486
 2      0000                    DATA  Segment       use16
 3      0000  4C 6F 72 65 6D    STRING          db      "Lorem"
 4      0005  7B                VAR0BD          db      123d
 5      0006  AB12              VAR0WH          dw      0ab12h
 6      0008  0006E6CC          VAR0DH          dd      06E6CCh
 7      000C                    DATA  ends
 8
 9      0000                    CODE1           Segment use16
10                              assume          ds:DATA, CS:CODE1
11      0000                    PROC1PROC far
12                              ; IMUL TESTS
13      0000  67| 6B 44 B2 06 0A IMUL ax, word ptr [edx         + esi *  4 + 6],  10
14      0006  66| 67| 6B 44 B2 06 +          IMUL eax, dword ptr [edx + esi         * 4 + 6], 10
15            0A
16      000D  67| 69 44 B2 06 03E8         IMUL ax, word ptr [edx         + esi *  4 + 6],  1000
17      0014  66| 67| 69 44  B2 06 +        IMUL eax, dword ptr [edx + esi         * 4 + 6], 33000
18            000080E8
19                              ; PUSHtests
20      001E  66| 67| FF 74  B2 06 push    dword ptr [edx+esi*4+6]
21      0024  67| FF 74 B2 06    push    word ptr [edx+esi*4+6]
22                              ; MOV tests
23      0029  67| C6 44 B2 06 0A MOV    byte ptr [edx + esi * 4 + 6], 10
24      002F  66| 67| 8B 44  B2 06 mov     eax, [edx+esi*4+6]
25      0035  66| 58           pop     eax
26      0037  CB                        ret
27      0038                    PROC1ENDP
28                                      ;
29                                      ; OR TESTS
30      0038  0B C3                     or ax,   bx
31
32      003A  66| 0B C3                or eax, ebx
33
34      003D  0A C3                     or al,   bl
35
36      003F  0B D8                     or bx,   ax ; Valid
37
```

```
38      0041  67| 8B 74 B2 06          mov     si, [edx+esi*4+6]
39      0046  74 02 90 90              jz      vol
40      004A                           vol:
41      004A                           CODE1           ENDS
42
43      0000                           CODE2           Segment use16
44                                     assume          ds:DATA, CS:CODE2
45      0000                           PROC2           PROC
46      0000  C3                       ret
47      0001                    PROC2          ENDP
48
49      0001  0A CC                    or      cl, ah
50      0003  E8 FFFA                  call    PROC2
51      0006  9A 00000000sr                    call    PROC1
52      000B                           CODE2           ENDS
53                                     END
```

Symbol Table

Symbol Name	Type	Value
??DATE	Text	"06/03/21"
??FILENAME	Text	"TEST1 "
??TIME	Text	"17:08:50"
??VERSION	Number	0205
@CPU	Text	0D1FH
@CURSEG	Text	CODE2
@FILENAME	Text	TEST1
@WORDSIZE	Text	2
PROC1	Far	CODE1:0000
PROC2	Near	CODE2:0000
STRING	Byte	DATA:0000
VAR0BD	Byte	DATA:0005
VAR0DH	Dword	DATA:0008
VAR0WH	Word	DATA:0006
VOL	Near	CODE1:004A

Groups & Segments	Bit	Size	Align	Combine	Class
CODE1	16	004A	Para	none	
CODE2	16	000B	Para	none	
DATA	16	000C	Para	none	