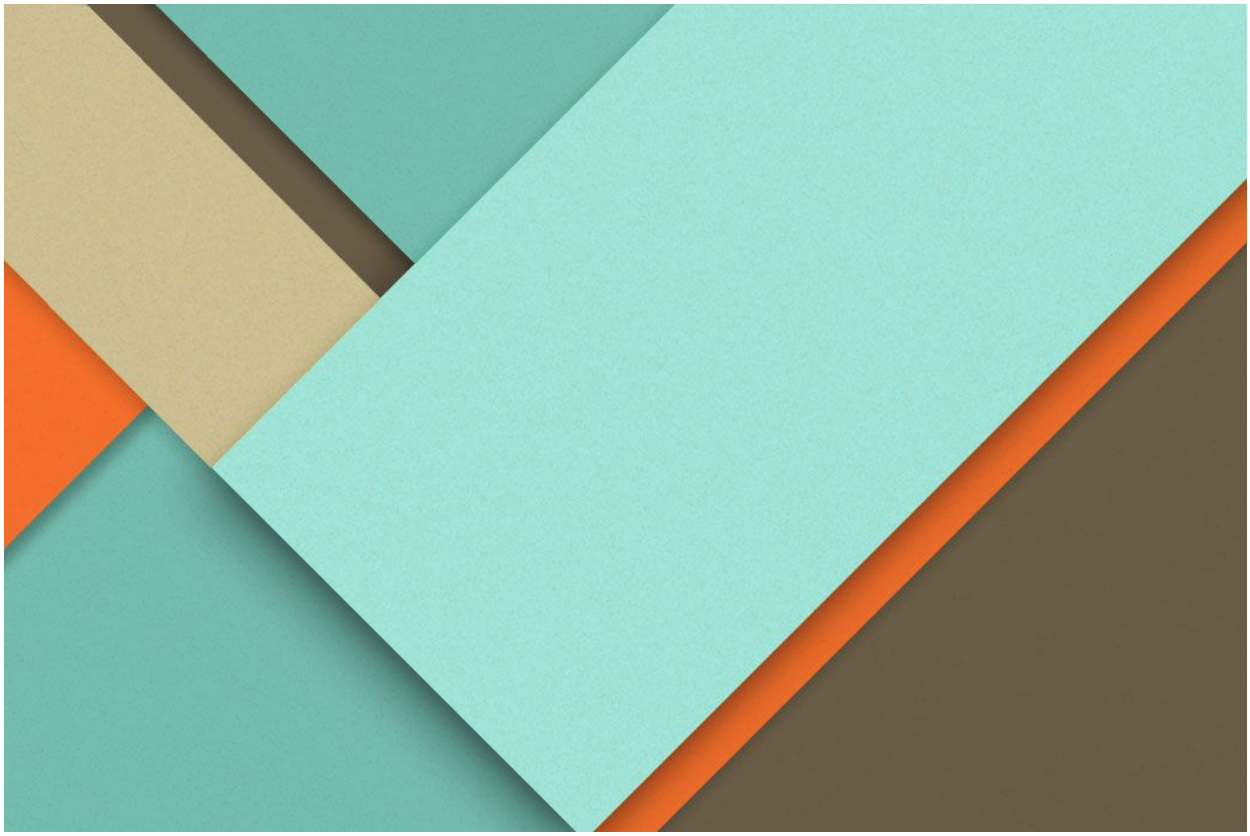

Two pass assembler

14/5/2019

group(7)

1. Mahmoud Hamed Sharshar(46)
2. Mahmoud Fathy Aboeleneen(47)
3. Karim Ibrahim Mostafa(30)
4. Abd_elmonem Badre el-sawy (26)

Overview

The two pass assembler performs two passes over the source program:

In the first pass, it reads the entire source program, looking only for label definitions. All the labels are collected, assigned address, and placed in the symbol table in this pass, no instructions as assembled and at the end the symbol table should contain all the labels defined in the program. To assign address to labels, the assembles maintains a Location Counter (LC).

In the second pass the instructions are again read and are assembled using the symbol table. Basically, the assembler goes through the program one line at a time, and generates machine code for that instruction. Then the assembler proceeds to the next instruction

Specifications

1. The assembler is to execute by entering: assemble < pass2 sourceFile>

2. The source file for the main program for this phase is to be named assemble.cpp

3. The output of the assembler should include (at least):

a) Object-code file

b) A report at the end of pass2. Pass1 and Pass2 errors included as part of the assembler report, exhibiting both the erroneous lines of source code and the error.

4. The assembler support:

a) EQU and ORG statements.

b) Simple expression evaluation. A simple expression includes simple (A B) operand arithmetic, where is one of +, -, *, / and no spaces surround the operation, eg. A+B.

Design:

- The program divided into three parts:

1.pass2.h⟶ contain main functions for pass two:

- **getObjectCode: calculate object code for each instruction**
- **calExpression : calculate value for expressions**
- **writeListFile : write result of pass two in list file**
- **writeTextRecord: write object program in objectFile**
- **excutePassTwo: iterate throw list file for assembling**

2.numberBases.h⟶ contain functions related to converting between different bases of number

- **convertToBase10 : convert any base to base 10**
- **convertFromBase10: convert from any base to base 10**
- **convertHexToBin: convert hexadecimal number to binary**
- **convertBinToHex:convert binary number to hexadecimal**
- **addHex:calculate the result of adding two hex digits**
- **twosComplement:calculate twos complement**
- **subHex:calculate the result of subtraction two hex digits**
- **mulHex:multiply two hex numbers**
- **divHex:division two hex numbers**
- **getStringWithLenght: get a string with specific length**
- **getOpcodeBin:calculate opCode in binary to specific format**
- **checkHex:check that number is a hexadecimal number**

3.assembler.cpp⟶ contains the main program that creates the object of the pass2 class to assemble specific file.

Main data structures :

LabelInfo structure: ○ Hold basic information of any label such as its address and flag to indicate an error.

- **opCodeInfo structure:**

  ○ Hold information of each operation code such as machine language equivalent code format, and the number of operands required for this opCode.

- **statParts structure:**

  ○ Represent parts of each statement in source file like a label, opcode, operand,comment and commentOnly that is a flag to indicate this is comment only or not.

- **Map data structure(built in):**

  ○ Store operation code as key and its information as value (opCodeTable).

  ○ Store label as key and its information as value(symbol table).

- **linkedlist:**

  ○ Store each instructions as pair of location counter and object code

- **expResult structure :**

  ○ Store result of expressions and their type : relative or absolute

Assumptions:

- If there is an error in pass one it will not execute pass two
- There is no empty line
- Simple expression only allowed to use
- Any comment must be preceded with the (.) character.

Algorithms description :

- **Steps of the pass2 algorithm:**
  - open list file for reading statement
  - Read first line and loop until hit first instruction.
  - If the first statement is (start statement), specify starting address to the operand.
  - Read the next instruction.
  - Look up the opcode in the optab and if exist, calculate object code for this instruction depending on its format and type of addressing.
  - write this line and its object code in list file .
  - If opcode equal end loop .
  - write object codes in object file .

Pass 2:

```
begin
   read first input line {from intermediate file}
   if OPCODE = 'START' then
      begin
         write listing line
         read next input line
      end {if START}
   write Header record to object program
   initialize first Text record
   while OPCODE ≠ 'END' do
      begin
         if this is not a comment line then
            begin
               search OPTAB for OPCODE
```

```
            if found then
               begin
                  if there is a symbol in OPERAND field then
                     begin
                        search SYMTAB for OPERAND
                        if found then
                           store symbol value as operand address
                        else
                           begin
                              store 0 as operand address
                              set error flag (undefined symbol)
                           end
                     end {if symbol}
                  else
                     store 0 as operand address
                  assemble the object code instruction
               end {if opcode found}
            else if OPCODE = 'BYTE' or 'WORD' then
               convert constant to object code
```

```
            if object code will not fit into the current Text record then
               begin
                  write Text record to object program
                  initialize new Text record
               end
            add object code to Text record
         end {if not comment}
      write listing line
      read next input line
   end {while not END}
   write last Text record to object program
   write End record to object program
   write last listing line
end {Pass 2}
```

Sample runs :

Example1: source File

```
STRCP2   START   1000
FIRST    STL     RETADDR
         LDT     #11
         LDX     #0
MOVECH   LDCH    STR1,X
         STCH    STR2,X
         TIXR    T
         JLT     MOVECH
         J       @RETADDR
STR1     BYTE    C'EOF'
STR2     RESB    11
RETADDR  RESW    1
END      FIRST
```

List File:

START PASS TWO

LineNumber	ADDRESS	LABEL	Mnemonic	OPERANDS	OBJECTCODE
1	1000	FIRST	STL	RETADDR	172022
2	1003		LDT	#11	75000B
3	1006		LDX	#0	50000
4	1009	MOVECH	LDCH	STR1,X	53A00B
5	100C		STCH	STR2,X	57A00B
6	100F		TIXR	T	B850
7	1011		JLT	MOVECH	3B2FF5
8	1014		J	@RETADDR	3E200E
9	1017	STR1	BYTE	C'EOF'	454F46
10	101A	STR2	RESB	11	
11	1025	RETADDR	RESW	1	
12	1028		END	FIRST	

Object File:

```
H STRCP2   001000 28
T 001000 C 172022 75000B 050000 53A00B 57A00B
T 00100F 8 00B850 3B2FF5 3E200E 454F46
E 001000
```

Exampl2(lecture Example):

Loc		Source statement		Object code
0000	COPY	START	0	
0000	FIRST	STL	RETADR	17202D
0003		LDB	#LENGTH	69202D
		BASE	LENGTH	
0006	CLOOP	+JSUB	RDREC	4B101036
000A		LDA	LENGTH	032026
000D		COMP	#0	290000
0010		JEQ	ENDFIL	332007
0013		+JSUB	WRREC	4B10105D
0017		J	CLOOP	3F2FEC
001A	ENDFIL	LDA	EOF	032010
001D		STA	BUFFER	0F2016
0020		LDA	#3	010003
0023		STA	LENGTH	0F200D
0026		+JSUB	WRREC	4B10105D
002A		J	@RETADR	3E2003
002D	EOF	BYTE	C'EOF'	454F46
0030	RETADR	RESW	1	
0033	LENGTH	RESW	1	
0036	BUFFER	RESB	4096	
	.			
	.	SUBROUTINE TO READ RECORD INTO BUFFER		
1036	RDREC	CLEAR	X	B410
1038		CLEAR	A	B400
103A		CLEAR	S	B440
103C		+LDY	#4096	75101000
1040	RLOOP	TD	INPUT	E32019
1043		JEQ	RLOOP	332FFA
1046		TD	INPUT	DB2013
1049		COMPR	A,S	A004
104B		JEQ	EXIT	332008
104E		STCH	BUFFER,X	57C003
1051		TIXR	T	B850
1053		JLT	RLOOP	3B2FEA
1056	EXIT	STX	LENGTH	134000
1059		RSUB		4F0000
105C	INPUT	BYTE	X'F1'	F1

```
.
.       SUBROUTINE TO READ RECORD INTO BUFFER

105D    WRREC   CLEAR   X       B410
105F            LDT     LENGTH  774000
1062    WLOOP   TD      OUTPUT  E32011
1065            JEQ     WLOOP   332FFA
1068            LDCH    BUFFER,X 53C003
106B            WD      OUTPUT  DF2008
106E            TIXR    T       B850
1070            JLT     WLOOP   3B2FEF
1073            RSUB            4F0000
1076    OUTPUT  BYTE    X'05'   05
                END     FIRST
```

List File :

```
START PASS TWO

LineNumber  ADDRESS  LABEL   Mnemonic  OPERANDS   OBJECTCODE
1           0        FIRST    STL       RETADR     17202D
2           3                 LDB       #LENGTH    69202D
3           6                 BASE      LENGTH
4           6        CLOOP    JSUB      RDREC      4B101036
5           A                 LDA       LENGTH     32026
6           D                 COMP      #0         290000
7           10                JEQ       FIL        332007
8           13                JSUB      WRREC      4B10105D
9           17                J         CLOOP      3F2FEC
10          1A       FIL      LDA       EOF        32010
11          1D                STA       BUFFER     F2016
12          20                LDA       #3         10003
13          23                STA       LENGTH     F200D
14          26                JSUB      WRREC      4B10105D
15          2A                J         @RETADR    3E2003
16          2D       EOF      BYTE      C'EOF'     454F46
17          30       RETADR   RESW      1
18          33       LENGTH   RESW      1
19          36       BUFFER   RESB      4096
20          1036     RDREC    CLEAR     X          B410
21          1038              CLEAR     A          B400
22          103A              CLEAR     S          B440
23          103C              LDT       #4096      74F01000
24          1040     RLOOP    TD        INPUT      E32019
25          1043              JEQ       RLOOP      332FFA
```

24	1040	RLOOP	TD	INPUT	E32019
25	1043		JEQ	RLOOP	332FFA
26	1046		TD	INPUT	E32013
27	1049		COMPR	A,S	A004
28	104B		JEQ	EXIT	332008
29	104E		STCH	BUFFER,X	57C003
30	1051		TIXR	T	B850
31	1053		JLT	RLOOP	3B2FEA
32	1056	EXIT	STX	LENGTH	134000
33	1059		RSUB		4F0000
34	105C	INPUT	BYTE	X'F1'	F1
35	105D	WRREC	CLEAR	X	B410
36	105F		LDT	LENGTH	774000
37	1062	WLOOP	TD	OUTPUT	E32011
38	1065		JEQ	WLOOP	332FFA
39	1068		LDCH	BUFFER,X	53C003
40	106B		WD	OUTPUT	DF2008
41	106E		TIXR	T	B850
42	1070		JLT	WLOOP	3B2FEF
43	1073		RSUB		4F0000
44	1076	OUTPUT	BYTE	X'05'	05
45	1077		END	FIRST	

Object File :

```
H COPY     000000 1077
T 000000 D 17202D 69202D 4B101036 032026 290000
T 000010 D 332007 4B10105D 3F2FEC 032010 0F2016
T 000020 D 010003 0F200D 4B10105D 3E2003 454F46
T 001036 A 00B410 00B400 00B440 74F01000 E32019
T 001043 B 332FFA E32013 00A004 332008 57C003
T 001051 B 00B850 3B2FEA 134000 4F0000 0000F1
T 00105D B 00B410 774000 E32011 332FFA 53C003
T 00106B B DF2008 00B850 3B2FEF 4F0000 000005
E 000000
```