System programming
Project: Phase1

Group # 11

Names:

1. **Esraa Mohamed Hashish (12)**
2. **Amira Mohammed Fathy (14)**
3. **Gehad Fathy Mohamed (19)**
4. **Rewan Alaa(23)**
5. **Safaa Hassan wally (34)**

Requirements specification:

{Implement Pass1 of the assembler }

- The pass1 is to execute by entering pass1<source-file-name>
- The program is capable of handling source lines that are
  - Comments
  - Directives (BYTE, WORD, RESW, and RESB, START and END).
  - Operations that are in opcode**.**
  - a directive that is not implemented should be ignored possibly with a warning .
- the parser is to be minimally capable of decoding 2, 3 and 4-byte instructions as follows:
  - 2-byte with 1 or 2 symbolic register reference (e.g., TIXR A, ADDR S,A)
  - RSUB (ignoring any operand or perhaps issuing a warning)
  - 3-byte including immediate, indirect, and indexed addressing
  - 4-byte absolute with symbolic or non-symbolic operand to include immediate, indirect, and indexed addressing.
- Hexadecimal addresses that would begin with 'A' through 'F' must start with a leading '0' to distinguish them from labels.
- The source program can be written using either uppercase or lowercase letters.
- The source program to be assembled can be free format
- If a source line contains "." in the first byte, the entire line is treated as a comment

Design:

Main Class:

- Takes from user source file path to be assembled, and it is considered to be free format then pass it to **Parser class** .

Parser Class:

- Parses free format files and generate the appreciate syntax errors.

- Steps:
  - Call **ReadFile class** to read source code file.
  - Call **OpcodeGenerator class** to generate optable.
  - then for each line in the source code :split it ignoring all white spaces and determine label , instruction and operand positions
  - validate operand and label fields and if not valid it set the appreciate error
  - Finally, it passes a vector to **PassOneAlgorithm class** consists of label, instruction, operand, format, error messages or if all the line is a comment, send it as the last element.

❖ PassOneAlgorithm Class:

- This main task of this class is to check logical errors
  - If no existing errors : Update location counter and at the last of pass generate symbol table.
  - Else: Location counter doesn't change at the last of pass print "Incomplete assembly"
- Types of errors:

- START :
  - Can be existing or not.
  - Can't be duplicate.
  - The operand must be existing and hexadecimal address.
- END
  - Must be existing
  - Must be last statement in source code
  - Can't have duplicate .
  - Can't have label
  - May have operand or not
- RESW &RESB :
  - The operand mut be decimal integer
- BYTE:
  - The operand must be
    a) X' ' with odd number of hexadecimal digits.
    b) C''with any characters
- WORD:
  - Operand can be integer or simple expression

• Data structures:

❖ ReadFile Class :

- vector <String> :
  - store source code lines generated by the file reader
  - listing all directives
  - listing file lines

❖ Parser Class:

- vector <vector<String>>
  - store source code statements generated by the Parser
- Map<String, Vector<String> > :
  - store optable → map each instruction to format and opcode

❖ PassOneAlgorithm:

- vector<string> lisiting_file
  - it is the returned vector from the function (take_source)
  - it consists of lines of source codes + location counter +errors
  - it is supposed to be printed in list file
- map<string,string> symbol_table
  - the label if is valid ,then it is added to this map with its address in memory in hex.
  - If source code has no errors it is also printed in listing file.

• Algorithm:

❖ Parser Class :

FOR each line in the source code

IF the line begins with a dot then it is a comment

ELSE

- split it on all white spaces
- search for the Mnemonic {Instruction or directive} position in the statement

```
            IF
                not found then set ERROR "InValid statement"
            ELSE
                determine the label ,  Mnemonic and operand positions
                according to Mnemonic position and the number of words
                    returned  from the splitting
                      IF the second word is a Mnemonic then the first is a label
                            then set it  in the label position  and if not valid  set
                            the ERROR  "Illegal label name"
                      ELSE
                            no label
                      END IF
                if the Mnemonic is from Directives that isn't supported yet
                then set  a WARNING "Unsupported Directive"
                set the format of the instruction
                check the Operand
                 IF there is an operand
                      - valid it
                      - check if the Mnemonic doesn't need the operand
                        {RSUB , LTORG , CSECT } then set the ERROR
                       to "extra characters at end of statement"
                      - set the operand field
                ELSE
                      - check if the Mnemonic need the operand {LDA ,
                      CPMPR , RESB , WORD , ... } then set the
                      ERROR to "missing operand field"

                END IF
            END IF
        END IF
END  FOR
```

- Valid label →
    ** label name shouldn't be (Mnemonic name, register name, start with numbers, contains symbols & % $….) ,
    ** length of the label doesn't exceeds the maximum length (MAX_LEN=7)
    if not valid set the ERROR to "**Illegal label name**"
- Valid Operand →

** the length of the operand doesn't exceeds the maximum length (MAX_LEN=18)
else generate an ERROR **"exceeds maximum operand length"**

** check if the label of **only** {BYTE,WORD}
- X'hexa value'
- C'characters'
- Numeric value
- Numeric values separated by commas.

if it isn't matching then set ERROR **"unrecognized operand"**

Format Setting →
- format → 4 : if instruction is format 3 and starts with +
- format → 2: if instruction is format 2 and starts with + and generate an ERROR **"Can not be format 4 instruction"**
- else according to the value of format stored in the optable

---

PassOneAlgorithm Class :

- **string form_line(vector<string> line)**
  - it form a line from a vector to be printed in the file and mainly depends on calculating spaces and sizes
- **string convert_to_string(int number)**
  -takes number and covert it to string in hexadecimal form. (using a built in function" ss << hex <<uppercase <<number")
- **bool is_number(string s)**

```
for I <= 0 to s.size()
        if(!isdigit(s.at(i)))
                return false;
return true;
```

- **int is_hex(string s)**

```
if(firstChar != 0)
```

```
            return -1;
for I <= 1 to s.size()
      if(s.at(i) is not from A TO F  )
                        return -1
convert to decimal using built in function(  str >> std::hex >> value)
 return value
```

- **vector<string> take_source(vector<vector<string> > source_file)**

```
 i <- 0 // to iterate on vector
 vector<string> line <- source_file[i];
 if(line is comment)
            lisiting_file.add(comment
 i++
line <- source_file[i];
end if
 if( no existing  error)
        if (operand = START ){
            i++;
           new_line <- form_line(line)
          lisiting_file.add(new_line)
         if(no opearnd)
                  ERROR
           else if(operand is number || (operand+”0”) ishex)
                    location_counter = is_hex("0"+operand);
                   start_address = location_counter;
          else
```

```
                ERROR
}
for I to   source_file.size() {
        line <- source_file[i]
       if(line is comment)
              lisiting_file.add(comment);
        new_line       <-form_line(line)
        lisiting_file.add(new_line)
        if(in valid instruction)
                ERROR
        if(exist label)
                If (invalid || in SYMTAB )
                        ERROR
        Else
              add it to symbol table
        if(operation == END){
                if(not last statement)
                        ERROR
                  else
                          exist_end <- true
                            If (exist an operand  &&not in symbol table || exist label)
                                        ERROR
                            program_lenght <- convert_to_string(location_counter-
                            start_address)
        }
        if(operation ==START  && ! first statement)
```

```
            ERROR
if(format is 2 || 3 || 4)
            location_counter  += location counter
if(!operand is existing || !valid)
            ERROR
else if(operation IS RESW or RESB or WORD){
        If (operation != WORD and operand is not number)
                ERROR
        else
            if(operation IS RESW)
                    location_counter += operand*3
              else if(operation IS RESB)
                    location_counter += operand
              else              // word
                      If (operand contains ‘,’)
                              Group = operand.Split(,)
                              location_counter += group.size()*3
                      else if (operand is existing && (valid ||number))
                              location_counter +=3
                      else
                              ERROR
        }
        else if(operation = BYTE){
                    if(first_char == 'X')
                          If (operand is not hex or length is odd)
                              ERROR
                    else
                        location_counter += lenght/2
                    Else if (first_char == 'C')
                     location_counter += lenght
              else    //not x or c
              ERROR
        }
```

```
  if(!exist_end)
     ERROR
if (!existError)
      print SYMTAB
else
      print    “INCOMPLETE ASSEMBLY”
```

Assumptions:

1) the format of the files is assumed to be free format and that is including also fixed format
2) no comment field inline with the statement
3) mnemonic is first or second word else statement isconsidered a comment
4) The word after every operation is operand
5) the operand after Start is a valid address .
6) the operand after END is a label in SYMTAB.
7) WORD takes only integer values or operand on the type integer,integer.
8) The Assembler doesn't handle expressions.
9) The Assembler doesn't handle operands unless the mnemonic is WORD ,RESW,BYTE ,RESB.
10) If defined as hexadecimal " x'hexa' " then hexa is even number of hexadecimal(14 max) && c ' chararters'(15 max)

Sample runs:

1.

ListingFile - Notepad

File Edit Format View Help

```
.234567890123456789012345678 90
000000 FJRJEOJFEPJFEPJFESTART    -0FFF
      **Illegal label name
000000           LDB      0FF
000003           LDS      #A111
000006           LDX      #GFFFF
000009           LDT      #-11111
00000C           LDA      NUM
00000F           JSUB     NEG
000012           JSUB     LEN
000015           JSUB     DIVISORRRRRRRRR
000018 JODD      SKDHS    JALDA
      **Invalid Statement
.********OPERATION********
000018           LDX      LENGTH
00001B           LDA      NUM
00001E OPR       DIVR     T,A
000020           ADD      #48
000023           JSUB     PRINT
000026           COMPR    T,S
000028           JEQ      FINAL
00002B           DIVR     B,T
```

source (1) - Notepad

File Edit Format View Help

```
.234567890123456789012345678 90
fjrjeojfepjfepjfe  START   -0fff
         LDB     0ff
         LDS     #a111
         LDX     #gfffff
         LDT     #-11111
         LDA     NUM
         JSUB    NEG
         JSUB    L  E  N
         JSUB    DI  V   IS      ORRRRRRRRR
jodd skdhs jalda
.********OPERATION********
         LDX     LENGTH
         LDA     NUM
OPR      DIVR    T,   A


         ADD     #                              48
         JSUB    P  RI  NT
         COMPR   T,S
         JEQ     FINAL
         DIVR    B,T
```

ListingFile - Notepad

File Edit Format View Help

```
00002B            DIVR     B,T
00002D            STA      NUM
000030            J        OPR
000033 FINAL      J        *
.********PRINT**************
000036 PRINT      TD       DEV
000039            JEQ      PRINT
00003C            WD       DEV
00003F            RSUB
000042 LEN        LDA      NUM
000045 LOOP       DIV      #10
000048            ADDR     S,X
00004A            COMP     #0
00004D            JGT      LOOP
000050            STX      LENGTH
000053            RSUB
000056 LOOP       RES      555
      **Invalid Statement
.********NEGATIVE********
000056 NEG        COMP     #0
000059            JLT      OUTPUT
00005C            RSUB
```

source (1) - Notepad

File Edit Format View Help

```
          DIVR     B,T
          STA      NUM
          J        OPR
FINAL     J        *
.********PRINT**************
PRINT     TD       DEV
          JEQ      PRIN  T
          WD       DEV
          RSUB
LEN       LDA      NUM
LOOP      DIV      #10
          ADDR     S,X
          COMP     #0
          JGT      L  O   OP
          STX      LENGTH
          RSUB
loop      res    555
.********NEGATIVE********
NEG       COMP     #0
          JLT      OUTPUT
          RSUB
OUTPUT    TD       DEV
```

ListingFile - Notepad

File Edit Format View Help

```
.*********DATA*************
000074 NUM        BYTE     X'800001'
000077 ONE        WORD     -0F55
      **unrecognized operand
000077            WORD
      **missing operand field
000077            WORD     1111
00007A            WORD     GGGGG
      **unrecognized operand
00007A            WORD     -57FFF
      **unrecognized operand
00007A            WORD     -5000
00007D            WORD     1,2,3,55
000089            WORD     FFF,55
      **unrecognized operand
000089 DEV        BYTE     X'045'
        ***odd length for hex string***
000089 DEV        BYTE     X'GFG'
      **unrecognized operand
000089 DEV1       BYTE     X'1FAB'
00008B LENGTH     RESW     1
00008E            END
```

source (1) - Notepad

File Edit Format View Help

```
          JEQ      OUTPUT
          MUL      ONE
          STA      NUM
          LDCH     #45
          WD       DEV
          RSUB
.*********DATA*************
NUM       byte     x'800001'
ONE       WORD     -0f55
          Word
Word    1111
word     ggggg
word    -57fff
word    -5000
word      1,2,3,55
word      fff,55
DEV       BYTE     X'045'
DEV       BYTE     X'GFG'
DEV1      BYTE     X'1FAB'
LENGTH    RESW     1
          END
```

2.

```
        START   0000
Q2      LDA     #0
        LDX     #1
        LDS     #10
        LDL     #0
AGAIN   TD      INDEV
        JEQ     AGAIN
        RD      INDEV
        COMP    #4
        JEQ     EXIT
        SUB     #48
        MULR    X,l
        ADDR    A,L
        MULR    S,X
        J       AGAIN
EXIT    RMO     L,A
        J       *
INDEV   BYTE    X'F3'
        END     Q2
```

```
000000          START   0000
000000 Q2       LDA     #0
000003          LDX     #1
000006          LDS     #10
000009          LDL     #0
00000C AGAIN    TD      INDEV
00000F          JEQ     AGAIN
000012          RD      INDEV
000015          COMP    #4
000018          JEQ     EXIT
00001B          SUB     #48
00001E          MULR    X,L
000020          ADDR    A,L
000022          MULR    S,X
000024          J       AGAIN
000027 EXIT     RMO     L,A
000029          J       *
00002C INDEV    BYTE    X'F3'
00002D          END     Q2

>>   e n d   o f   p a s s   1
```

```
>>   e n d   o f   p a s s   1

>>   ********************************
```

3.

```
>>        s y m b o l     t a b l e
          name          value
 ---------------------------------
          AGAIN           C
          EXIT            27
          INDEV           2C
          Q2              0
```

3.

source (2) - Notepad

File Edit Format View Help

```
          START   0000
    start   1000
1fff        LDA     #0
f&^54          LDX     #1
          LDS     #10
          LDL     #0
AGAIN     TD      INDEV
          JEQ     AGAIN
          RD      INDEV
          COMP    #4
          JEQ     EXIT
start  1000
          SUB     #48
          MULR    X,l
          ADDR
lda    length
start   label

end  55
          MULR    S,X
          J       AGAIN
EXIT      RMO     L,A
```

ListingFile (1) - Notepad

File Edit Format View Help

```
000000            START    0000
000000            START    1000
        ***START statement can't be preceded with instr
000000 1FFF       LDA      #0
      **Illegal label name
000000 F&^54      LDX      #1
      **Illegal label name
000000            LDS      #10
000003            LDL      #0
000006 AGAIN      TD       INDEV
000009            JEQ      AGAIN
00000C            RD       INDEV
00000F            COMP     #4
000012            JEQ      EXIT
000015            START    1000
        ***START statement can't be preceded with instr
000015            SUB      #48
000018            MULR     X,L
00001A            ADDR
      **missing operand field
00001A            LDA      LENGTH
00001D            START    LABEL
```

```
lda    length
start   label

end  55
          MULR    S,X
          J       AGAIN
EXIT      RMO     L,A
          J       *
INDEV     BYTE    X'F3'
          END     Q2
```

```
00001D            END      55
        ***end statement can't be followed by line*
00001D            MULR     S,X
00001F            J        AGAIN
000022 EXIT       RMO      L,A
000024            J        *
000027 INDEV      BYTE     X'F3'
000028            END      Q2
        ***illegal operand***
           Incomplete assembly
```

4.

ListingFile - Notepad

```
000000            START    FFF1
        ***illegal operand field***
000000 1FFF       LDA      #0
      **Illegal label name
000000 F&^54      LDX      #1
      **Illegal label name
000000            LDS      #10
000003            LDL      #0
000006 AGAIN      TD       INDEV
000009            JEQ      AGAIN
00000C            RD       INDEV
00000F            COMP     #4
000012            JEQ      EXIT
000015            SUB      #48
000018            MULR     X,L
00001A            ADDR
      **missing operand field
00001A            LDA      LENGTH
00001D            START    LABEL
        ***START statement can't be preceded with instr
00001D            MULR     S,X
00001F            J        AGAIN
```

source (1) - Notepad

```
          START    fff1
1fff        LDA      #0
f&^54          LDX      #1
          LDS      #10
          LDL      #0
AGAIN     TD       INDEV
          JEQ      AGAIN
          RD       INDEV
          COMP     #4
          JEQ      EXIT

          SUB      #48
          MULR     X,l
          ADDR
lda     length
start    label

          MULR     S ,                    X
          J        AGAIN
EXIT      RMO      L,A
          J        *
INDEV     BYTE     X'F31'
```

```
00001D            START    LABEL
        ***START statement can't be preceded w
00001D            MULR     S,X
00001F            J        AGAIN
000022 EXIT       RMO      L,A
000024            J        *
000027 INDEV      BYTE     X'F31'
        ***odd length for hex string***
000027            END      1000
        ***illegal operand***
           Incomplete assembly
```

```
          ADDR
lda     length
start    label

          MULR     S ,                    X
          J        AGAIN
EXIT      RMO      L,A
          J        *
INDEV     BYTE     X'F31'

          END      1000
```

5.

ListingFile - Notepad

File Edit Format View Help

```
000000            START    0FFF
000FFF            EQU      1000
**    Warning:Unsupported Directive  **
000FFF            EQU
**    Warning:Unsupported Directive  **
000FFF LABEL      EQU      50
**    Warning:Unsupported Directive  **
000FFF HERE       EQU
**    Warning:Unsupported Directive  **
000FFF            ORG      100
**    Warning:Unsupported Directive  **
000FFF            ORG
**    Warning:Unsupported Directive  **
000FFF PP         ORG      10
**    Warning:Unsupported Directive  **
000FFF PP         ORG
**    Warning:Unsupported Directive  **
.HERE WE CAN TEST
000FFF            END


>>    e n d    o f    p a s s    1

>>   ***************************************************

>>         s y m b o l     t a b l e
           name          value
 ---------------------------------
```

source - Notepad

File Edit Format View Help

```
          START    0fff
equ 1000
equ
label equ 50
here   equ
org   100
org
pp org 10
pp org
.here we can test
          END
```