



## SIC/XE Assembler

Producing code for the absolute loader used in the SIC/XE programming assignments using One Pass Assembler.

### Names:

Aya Gamal Mohamed	1
Enas Morsy Mohamed	20
Sara Mohamed Youssef	31
Linh Ahmed	50
Nada Fathy Ali	68

---

## Specifications

- Build a parser that is capable of handling source lines that are instructions, storage declaration, comments, and assembler directives.
- For instructions, the parser is to minimally be 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 PC-relative with symbolic operand to include immediate, indirect, and indexed addressing
  - 3-byte absolute with non-symbolic operand to include immediate, indirect, and indexed addressing
  - 4-byte absolute with symbolic or non-symbolic operand to include immediate, indirect, and indexed addressing
- The parser is to handle all storage directives (BYTE, WORD, RESW, and RESB).
- The assembler should support:
  - EQU and ORG statements.
  - 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.

## Main data structures

- vector

```
obj
```

 table: which contains all the instructions with their locctr and objectCode .
- Map<string,symbol\_info>:SYMTAB: which represents the symbol table.

```
struct symbol_info {  
    string address;  
    string flag="";  
    vector<std::string> reff;  
};  
  
struct preobj {  
    string Label = "";  
    string Operator = "";  
    string Operand = "";  
    string objectCode = "";  
    string Opcode = "";  
    int Format = 0;  
    string locctr = "0";  
};
```

## Design

- [operations.h/operations.cpp](#)

Contain functions to get format,opCode and no. of operand of each operator.

- [structures.h/structures.cpp](#)

Contain the two main structures in the program.

- [Conversions.h/Conversions.cpp/Convert.h/Convert.cpp](#)

Contain all the conversions used in the program

Ex: from bin to Hex,from dec to Hex,.....etc

---

- **ObjectCode.h/ObjectCode.cpp**

Contain functions

- to store registers and labels with their address in the symbol table.
- to form the object code using the symbol table and by calling the Formats class.

- **Formats.h/Formats.cpp**

Contains functions to form object code of format 3 or 4 by calculating its displacement and the nixbpe.

- **WriteFile.h/WriteFile.cpp**

To write the object file using the symbol table and Table map.

## Algorithms description

The algorithm is to implement one pass assembler which translates mnemonic operation codes to their machine language equivalents.

- Read the file line by line.
- For each line check its validation and print error if a wrong line is entered.
- If the line is correct, remove extra spaces and split the instruction to its label, operator and operand and store them.
- Check if the instruction is the START one save the program name and set the locctr to the operand.

- 
- Check if the instruction is one from the storage directives(WORD,RESW,...) ,store the label in the symbol table and increment the locctr according to its case .
  - Otherwise get the format and opcode of the operator and form the objectCode and add format to the locctr.
  - Check if all the labels are defined ,write the object file .Otherwise,print error message.

## Assumptions

- If the line is the EQU instruction the locctr doesn't change."as not mentioned in the lectures or text book"
- If the label is used before it was defined ,put 0000/000000 in the objectCode and when the label is found write in the object file each modification should be done.

## Sample runs.

- Not allow to use immediate addressing and indexing at the same time.

Microsoft Visual Studio Debug Console

```

S      4
SW     9
T      5
X      1
A      0
B      3
F      6
L      2
PC     8
S      4
SW     9
T      5
X      1
-----
A      0
B      3
F      6
L      2
PC     8
S      4
SW     9
T      5
X      1
-----
error in operand
operator " LDA " doesn't match with operand " #U,X "

D:\MyAssembler2\Debug\MyAssembler2.exe (process 21048) e
Press any key to close this window . . .

```

assembler - Notepad

```

File Edit Format View Help
. copy 8 integers from u to v
. loop index <> 1
prog start 0000
bgn lds #3
    ldt #24
    ldx #0
loop lda #u,x
    sta v,x
    addr s,x
    compr x,t

.
. list of values for u
u word 1,2,3
  word 4
  word 5
  word 6
  word 7
  word 8
. array to store results
v resw 8
  end bgn

```

- Number of operands of some operators such as ADDR must be 2.

Microsoft Visual Studio Debug Console

```

B      3
F      6
L      2
LOOP   6   R
PC     8
S      4
SW     9
T      5
U      *   6
X      1
-----
A      0
B      3
F      6
L      2
LOOP   6   R
PC     8
S      4
SW     9
T      5
U      *   6
V      *   9
X      1
-----
error in operand
operator " ADDR " doesn't match with operand " S "
you shouldnt use any label not declared
D:\MyAssembler2\Debug\MyAssembler2.exe (process 19976)
Press any key to close this window . . .

```

assembler - Notepad

```

File Edit Format View Help
. copy 8 integers from u to v
. loop index <> 1
prog start 0000
bgn lds #3
    ldt #24
    ldx #0
loop lda u,x
    sta v,x
    addr s
    compr x,t

.
. list of values for u
u word 1,2,3
  word 4
  word 5
  word 6
  word 7
  word 8
. array to store results
v resw 8
  end bgn

```

Format 1 not supported.

Microsoft Visual Studio Debug Console

T	5	
X	1	
-----		
A	0	
B	3	
F	6	
L	2	
PC	8	
S	4	
SW	9	
T	5	
X	1	
-----		
A	0	
B	3	
F	6	
L	2	
LOOP	6	R
PC	8	
S	4	
SW	9	
T	5	
U	*	6
X	1	
-----		

Format 1 not supported  
you shouldnt use any label not declared  
D:\MyAssembler2\Debug\MyAssembler2.exe (process 4332) e  
Press any key to close this window . . .

assembler - Notepad

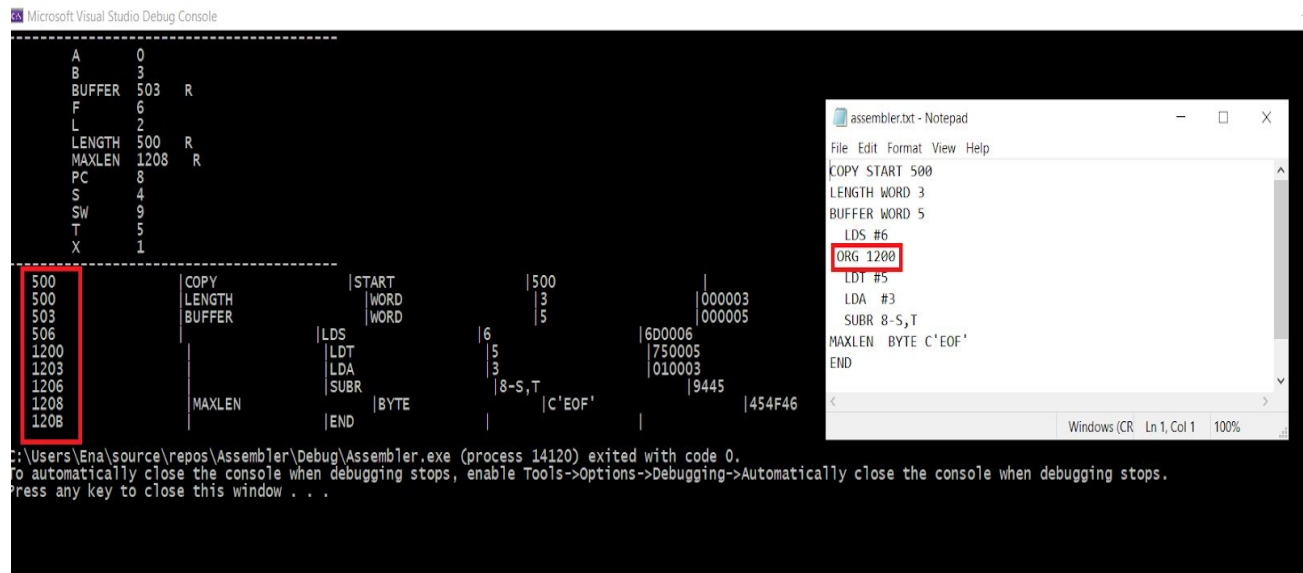
File Edit Format View Help

```
. copy 8 integers from u to v
. loop index <> 1
prog start 0000
bgn lds #3
    ldt #24
    ldx #0
loop lda u
    fix
    sta v,x
    addr s
    compr x,t
.
. list of values for u
u word 1,2,3
  word 4
  word 5
  word 6
  word 7
  word 8
. array to store results
v resw 8
  end bgn
```

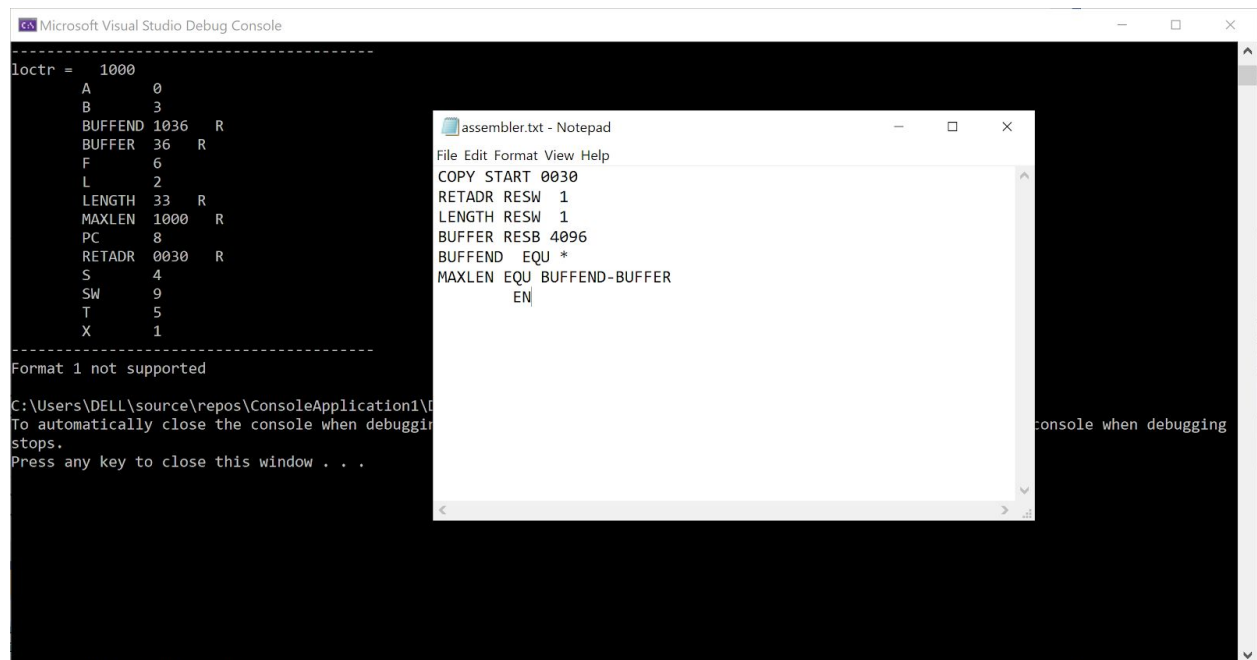


## Symbol statement

- ORG



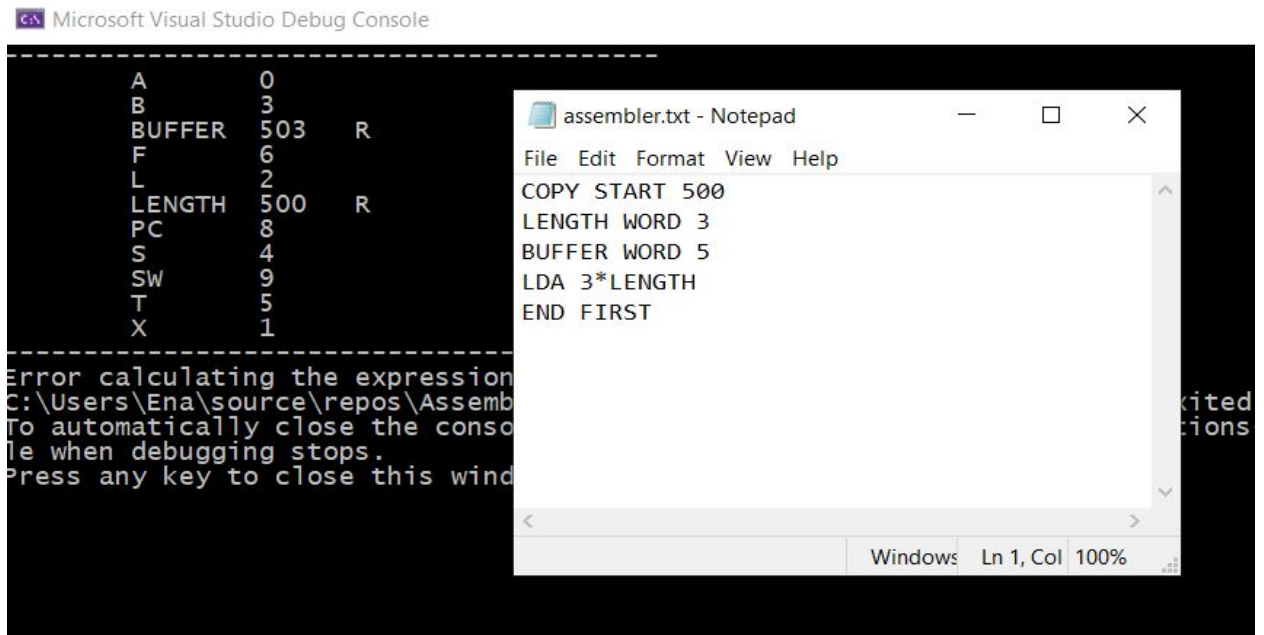
- EQU



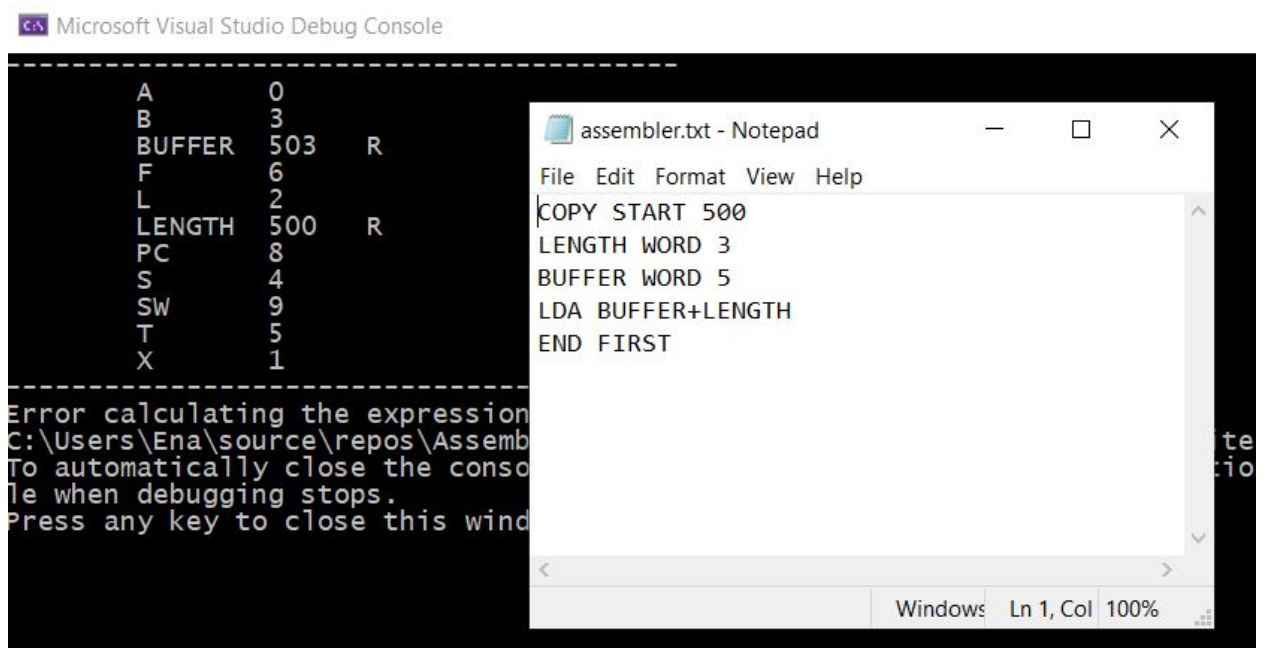


## Expressions:

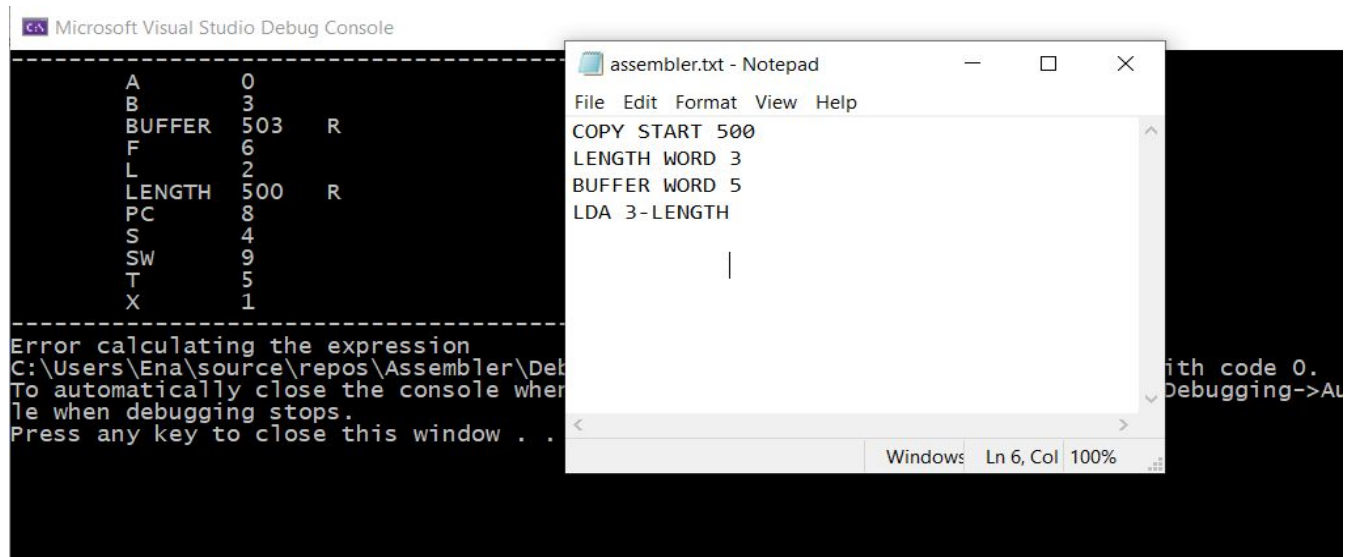
- Not allowed to use relative in multiplication or division.



- Relative+relative =ERROR



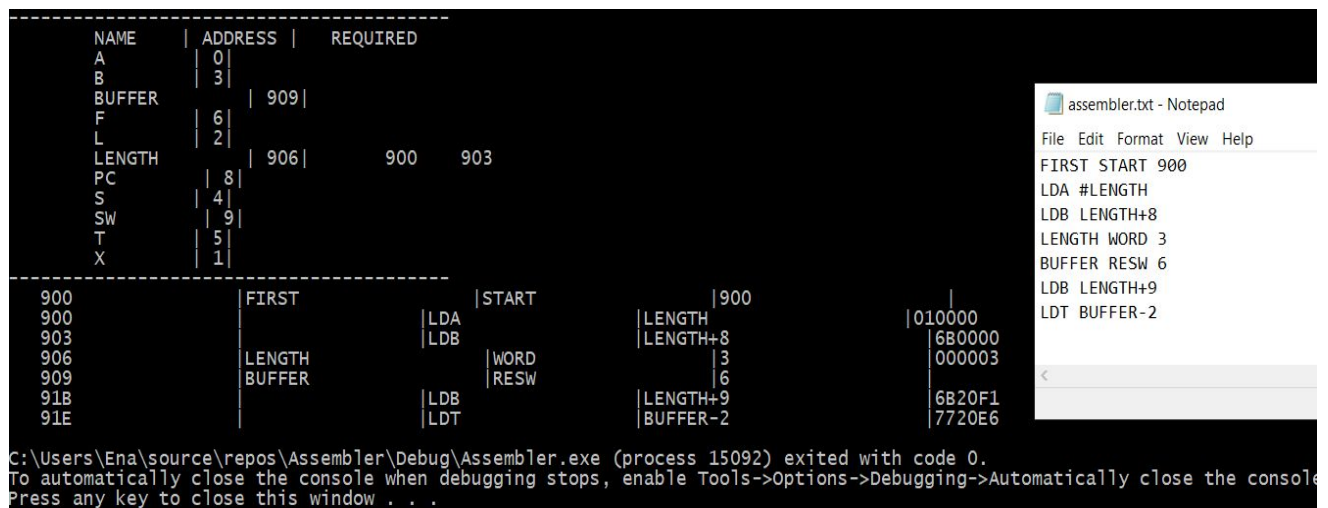
## ➤ Absolute-Relative=ERROR



## ➤ Right expression.

If the label is not defined, put zeros as in line 2,3.

Otherwise calculate it as in line 6,7.



## UNDEFINED LABEL:

Cloop is used in line 7 but the program is ended without defining it.

Microsoft Visual Studio Debug Console

```

-----
A      0
B      3
BUFFER 503 R
CLOOP  * 50E
F      6
L      2
LENGTH 500 R
MAXLEN 511 R
PC      8
S      4
SW      9
T      5
X      1
-----
you shouldnt use any label not declared
C:\Users\Ena\source\repos\Assembler\Debug\Ass
To automatically close the console when debug
le when debugging stops.
Press any key to close this window . . .

```

assembler.txt - Notepad

```

File Edit Format View Help
COPY START 500
LENGTH WORD 3
BUFFER WORD 5
LDA      #4
LDS      #6
COMPR    A,S
JEQ      CLOOP
MAXLEN   BYTE C'EOF'
END      FIRST

```

Windows Ln 1, Col 100%

## REDEFINED LABEL: LABEL is redefined.

Microsoft Visual Studio Debug Console

```

-----
variable is Found      NAME      | ADDRESS |  REQUIRED
A      0
B      3
F      6
L      2
LABEL  | 909 |      903
LENGTH | 906 |      900
PC      8
S      4
SW      9
T      5
X      1
-----
ERROR! redefined Label
C:\Users\Ena\source\repos\Assembler\Debug\Assembler.exe (process 8720) exited with code
To automatically close the console when debugging stops, enable Tools->Options->Debuggi
Press any key to close this window . . .

```

assembler.txt - Notepad

```

File Edit Format View Help
FIRST START 900
LDA #LENGTH
STA LABEL
LENGTH WORD 3
LABEL RESW 6
LABEL RESB 4000
END

```

Windows Ln 1, Col 100%

## Sample Run on Format2:

```

A 0
B 3
BUFFER 503 R
F 6
L 2
LENGTH 500 R
MAXLEN 511 R
PC 8
S 4
SW 9
T 5
X 1
-----
500 COPY START 500 500 000003
500 LENGTH WORD 3 000005
503 BUFFER WORD 5
506 LDS #6 6D0006
509 LDT #5 750005
50C LDA #3 010003
50F COMPR A,S+T A009
511 MAXLEN BYTE C'EOF' 454F46
514 END
-----
C:\Users\Ena\source\repos\Assembler\Debug\Assembler.exe (process 15208) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

assembler.txt - Notepad

```

File Edit Format View Help
COPY START 500
LENGTH WORD 3
BUFFER WORD 5
LDS #6
LDT #5
LDA #3
COMPR A,S+T
MAXLEN BYTE C'EOF'
END

```

myfile.txt - Notepad

```

File Edit Format View Help
H COPY 00500 000015
T 00500 14 000003 000005 6D0006 750005 010003 A009 454F46
E 00500

```

```

A 0
B 3
BUFFER 503 R
F 6
L 2
LENGTH 500 R
MAXLEN 511 R
PC 8
S 4
SW 9
T 5
X 1
-----
500 COPY START 500 500 000003
500 LENGTH WORD 3 000005
503 BUFFER WORD 5
506 LDS #6 6D0006
509 LDT #5 750005
50C LDA #3 010003
50F SUBR 8-S,T 9445
511 MAXLEN BYTE C'EOF' 454F46
514 END
-----
C:\Users\Ena\source\repos\Assembler\Debug\Assembler.exe (process 11160) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

assembler.txt - Notepad

```

File Edit Format View Help
COPY START 500
LENGTH WORD 3
BUFFER WORD 5
LDS #6
LDT #5
LDA #3
SUBR 8-S,T
MAXLEN BYTE C'EOF'
END

```

myfile.txt - Notepad

```

File Edit Format View Help
H COPY 00500 000015
T 00500 14 000003 000005 6D0006 750005 010003 9445 454F46
E 00500

```

## CORRECT SAMPLE RUNS:

Microsoft Visual Studio Debug Console

NAME	ADDRESS	REQUIRED
A	0	
B	3	
F	6	
L	2	
LOOP	1014	100B
LOP	101B	1011
PC	8	
S	4	
SW	9	
T	5	
THREE	1020	1005
X	1	

ADDRESS	OPCODE	INSTR	DATA	COMMENT
1000	COP	START		
1000		STL	10	
1003		RMO	L,A	
1005		LDA	THREE	
1008		COMP	4	
100B		JLT	LOOP	
100E		COMP	0	
1011		JEQ	LOP	
1014	LOOP	LDA	4	
1017		+ADD	9	
101B	LOP	LDX	1	
101E		RMO	X,A	
1020	THREE	WORD	3	
1023		END	1000	

```

COP START 1000
STL #10
RMO L,A
LDA THREE
COMP #4
JLT LOOP
COMP #0
JEQ LOP
LOOP LDA #4
+ADD #9
LOP LDX #1
RMO X,A
THREE WORD 3
END 1000
  
```

```

H COP 001000 000024
T 001000 13 15000A AC20 030000 290004 3B0000 2900 330000
T 00100B 02 1014
T 001014 07 010004 19100009
T 001011 02 101B
T 00101B 05 050001 AC10
T 001005 02 1020
T 001020 03 000003
E 001000
  
```