CSN 252

Tutorial-8

Rahul Kurkure
20114079
CSE O3

Aim : implementation of 2 pass SIC/XE assembler

The designed assembler contains all SIC / XE instructions and supports all four formats.

(1, 2, 3, 4) and addressing mode. It also supports all machine-independent features

1. literal
2. statements for defining symbols
3. expressions
4. Program blocks

Procedure for compilation and running the code

1. In the terminal run 'g++ pass2.cpp -o run' this command, and it will give us an executable file.
2. Now enter './run' to execute the newly generated file from (1)
3. After running this file, it will request for an user input which is the name of the file containing the assembly code of the input program
4. After this step 4 files will be created, object_sample.txt, intermediate_sample.txt, listing_sample.txt and error_sample.txt .

Assembler Working :

Pass1: - The intermediate and mistakess report are updated. If the supply report isn't always opened otherwise if the

intermediate report doesn't open, the corresponding mistakess is written withinside the mistakess report and if mistakess report

doesn't open, we print it to console. Then the primary line is taken as enter, test if it's miles a remark line.

Until the strains are comments, we take them as enter and print them to our intermediate report and

replace our line range. Once, the road isn't always a remark we test if the opcode is 'START', if located,

we replace the road range, LOCCTR and begin cope with if now no longer located, we initialize begin cope with and

LOCCTR as 0. Then, we use nested while() loops, wherein the outer loop iterates until opcode

equals 'END' and the internal loop iterates until, we get our opcode as 'END'. Inside the internal loop, we

test if line is a remark. If remark, we print it to our intermediate report, replace line range and

take withinside the subsequent enter line. If now no longer a remark, we test if there's a label withinside the line, if gift we

test if it's miles gift withinside the SYMTAB, if located we print mistakess saying 'Duplicate image' withinside the mistakess report

otherwise assign name, cope with and different required values to the image and shop it withinside the SYMTAB.

Then, we test if opcode is gift withinside the OPTAB, if gift we discover its layout after which

consequently increment the LOCCTR. If now no longer located in OPTAB, we test it with different opcodes like

'WORD', 'RESW', 'BYTE', 'RESBYTE', 'LTORG', 'ORG, 'BASE', 'USE' or 'EQU'. Accordingly, we insert the

symbols withinside the SYMTAB. For instance, for opcodes like USE, we insert a brand new BLOCK access withinside the

BLOCK map as described withinside the utility.cpp report, for LTORG we name the handle_LTORG() feature described

in pass1.cpp, for 'ORG', we factor out LOCCTR to the operand price given, for EQU , we test if

whether or not the operand is an expression then we test whether or not the expression is legitimate through the usage of the

evaluate_[removed]) feature, if legitimate we input the symbols withinside the SYMTAB. And if the opcode doesn’t suit with the above given opcodes, we print an mistakess message withinside the mistakess report.

Accordingly, we then replace our facts that is to be written withinside the intermediate report. After the loop

ends, we shop this system duration after which pass on for printing the SYMTAB, LITTAB and different tables.

After that we flow directly to the pass2().

handle_LTORG()- It makes use of byskip through reference. We print the literal pool gift until time through taking the

arguments from the pass1() feature. We run an iterator to print all of the literals gift withinside the LITTAB

after which replace the road range. If for a few literal, we did now no longer locate the cope with, we shop the

gift cope with withinside the LITTAB after which increment the LOCCTR on the idea of literal gift.

Evaluate_[removed])- It makes use of byskip through reference. We use some time loop to get the symbols from the

expression. If the image isn't always located withinside the SYMTAB, we hold the mistake message withinside the mistakess report.

We use a variable pairCount which continues the account of whether or not the expression is absolute or

relative and if the pairCount offers a few sudden price, we print an mistakess message.

Tables :

It includes all of the information systems required for our assembler to run. It includes the structs

for labels, opcode, literal and blocks. Maps are described for diverse tables with their indices as strings

with the names of the labels or opcodes as required.

The records for symbols are saved in map of struct Label namely, SYMTAB, records for

literals are saved in LITTAB, that for the OpCodes are saved withinside the OPTAB and that for the registers

are saved withinside the REGTAB. The records for the blocks are saved withinside the BLOCKS.

SYMTAB :

The struct carries facts of labels like name, address, block number, a person representing whether or not the label exits withinside the image desk or not, an integer representing whether or not label is relative

or not

OPTAB:

The struct incorporates facts of opcode like name, format, a person representing whether or not the

opcode is legitimate or not.

LITTAB:

The struct incorporates facts of literals like its price, deal with, block number, a person

representing whether or not the literal exits withinside the literal desk or not.

REGTAB:

The struct incorporates facts of registers like its numeric equivalent, a person representing

whether or not the registers exits or not.

BLOCKS:

The struct incorporates facts of blocks like its name, begin deal with, block number, vicinity counter

price for stop deal with of block, a person representing whether or not the block exists or not.

UTILITY:

UTILITYContains a few capabilities which might be used pretty frequently in different files.

getString()- takes in enter as a individual and returns a string.

intToStringHex()- takes in enter as int after which converts it into its hexadecimal equal with string

facts type.

expandString()- expands the enter string to the given enter size. It takes withinside the string to be expanded

as parameter and period of output string and the individual to be inserted on the way to amplify that

string.

stringHexToInt()- converts the hexadecimal string to integer and returns the integer fee.

stringToHexString()- takes in string as enter after which converts the string into its hexadecimal

equal after which returns the equal as string.

checkWhiteSpace()- exams if blanks are gift. If gift, returns genuine otherwise false.

checkCommentLine()- take a look at the remark via way of means of searching at the primary individual of the enter string, and

then for this reason returns genuine if remark otherwise false.

if_all_num()- exams if all of the factors of the string of the enter string are range digits.

readFirstNonWhiteSpace()- takes withinside the string and iterates till it receives the primary non-spaced

individual. It is a byskip via way of means of reference characteristic which updates the index of the enter string till the

clean area characters cease and returns void.

writeToFile()- takes withinside the call of the record and the string to be written directly to the record. Then writes the

enter string onto the brand new line of the record.

getRealOpcode()- for opcodes of layout 4, for example +JSUB the characteristic will see whether or not if the

opcode carries a few extra bit like ‘+’ or a few different flag bits, then it returns the opcode

leaving the primary flag bit.

getFlagFormat()- returns the flag bit if gift withinside the enter string otherwise it returns null string.

Class EvaluateString – carries the capabilities :

-peek()- returns the fee at the prevailing index.

-get()- returns the fee on the given index after which increments the index via way of means of one.

-range()- returns the fee of the enter string in integer layout.

Pass2

We take withinside the intermediate document as enter the usage of the readIntermediateFile() characteristic and

generate the list document and the item software. Similar to pass1, if the intermediate document is not able to

open, we are able to print the mistake message withinside the mistakess document. Same with the item document if not able to open.

We then study the primary line of the intermediate document. Until the traces are comments, we take them as

enter and print them to our intermediate document and replace our line variety. If we get opcode as

‘START’, we initialize out begin deal with because the LOCCTR, and write the road into the list document. We then

write the primary header file withinside the item software. Then till the opcode comes as ‘END’, we take

withinside the enter traces from the intermediate document after which replace the list document after which write the item

software withinside the textual content file the usage of the textrecord() characteristic. We will write the item code at the basis

of the styles of codecs used withinside the practise. Based on extraordinary styles of opcodes such as

‘BYTE’,’WORD’,’BASE’,’NOBASE’, we are able to generate extraordinary styles of item codes. For the layout three

and layout four practise layout, we are able to use the createObjectCodeFormat34() characteristic withinside the

pass2.cpp. For writing the give up file, we use the writeEndRecord() characteristic. For the instructions

with on the spot addressing, we are able to write the change file. When the internal loop for the

manipulate segment finishes, we are able to once more loop to print the following segment till the ultimate opcode for ‘END’

occurs.

readTillTab()- takes withinside the string as enter and reads the string till tab(‘t’) occurs.

readIntermediateFile()- takes in line variety, LOCCTR, opcode, operand, label and enter output files.

If the road is remark returns authentic and takes withinside the subsequent enter line. Then the usage of the readTillTab()

characteristic, it reads the label, opcode, operand and the remark. Based at the extraordinary styles of

opcodes, it'll rely withinside the vital situations to take withinside the operand.

createObjectCodeFormat34()- When we get our layout for the opcode as three or four, we name this

characteristic. It assessments the diverse conditions wherein the opcode may be after which taking into

attention the operand and the variety of 1/2 of bytes calculates the item code for the

practise. It additionally modifies themodification file while there may be a want to do so.

writeEndRecord()- It will write the give up file for the software.

After the execution of the pass1.cpp, we are able to print the Tables like SYMTAB, LITTAB, etc., in a separate

document after which execute the pass2.cppPass2: - We take withinside the intermediate document as enter the usage of the readIntermediateFile() characteristic and generate the list document and the item software. Similar to pass1, if the intermediate document is not able to

open, we are able to print the mistake message withinside the mistakess document. Same with the item document if not able to open.

We then study the primary line of the intermediate document. Until the traces are comments, we take them as

enter and print them to our intermediate document and replace our line variety. If we get opcode as

‘START’, we initialize out begin deal with because the LOCCTR, and write the road into the list document. We then

write the primary header file withinside the item software. Then till the opcode comes as ‘END’, we take

withinside the enter traces from the intermediate document after which replace the list document after which write the item

software withinside the textual content file the usage of the textrecord() characteristic. We will write the item code at the basis

of the styles of codecs used withinside the practise. Based on extraordinary styles of opcodes such as

‘BYTE’,’WORD’,’BASE’,’NOBASE’, we are able to generate extraordinary styles of item codes. For the layout three

and layout four practise layout, we are able to use the createObjectCodeFormat34() characteristic withinside the

pass2.cpp. For writing the give up file, we use the writeEndRecord() characteristic. For the instructions

with on the spot addressing, we are able to write the change file. When the internal loop for the

manipulate segment finishes, we are able to once more loop to print the following segment till the ultimate opcode for ‘END’

occurs.

readTillTab()- takes withinside the string as enter and reads the string till tab(‘t’) occurs.

readIntermediateFile()- takes in line variety, LOCCTR, opcode, operand, label and enter output files.

If the road is remark returns authentic and takes withinside the subsequent enter line. Then the usage of the readTillTab()

characteristic, it reads the label, opcode, operand and the remark. Based at the extraordinary styles of

opcodes, it'll rely withinside the vital situations to take withinside the operand.

createObjectCodeFormat34()- When we get our layout for the opcode as three or four, we name this

characteristic. It assessments the diverse conditions wherein the opcode may be after which taking into attention the operand and the variety of 1/2 of bytes calculates the item code for the

practise. It additionally modifies themodification file while there may be a want to do so.

writeEndRecord()- It will write the give up file for the software.

After the execution of the pass1.cpp, we are able to print the Tables like SYMTAB, LITTAB, etc., in a separate

document after which execute the pass2.cpp

Executing tbe below sample code :

```
SUM    START  0
FIRST   LDX    #0
        LDA    #0
        +LDB   #0
        +LDB   #TABLE2
        BASE   TABLE2
LOOP    ADD    TABLE,X
        ADD    TABLE2,X
        TIX    COUNT
        JLT    LOOP
        +STA   TOTAL
        RSUB
COUNT   RESW   1
TABLE   RESW   2000
TABLE2  RESW   2000
TOTAL   RESW   2
        END    FIRST
```

Object Code Generated

```
H^SUM ^000000^002F0A
T^000000^1E^050000010000691000006910179 41BA0131BC0002F200A3B2FF40F 102F04
T^00001E^03^4F0000

 M^00000B^05

M^00001B^05

E^000000
```