Objective

The objective of this programming project is to gain experience writing assembly language code using indexed addressing modes.

Background

Your assembly language program for this project will work with a linked list that is defined in a C file. Each node of the linked list contains data about an e-book. The data type for a node is declared as:

```
typedef struct book {
   char title[48] ;
   char author[48] ; // first author
   double price ;   // Price of book in $
   unsigned int year ; // year of e-book release
   struct book *next ;
} book ;
```

In memory, each struct book occupies an unbroken block of memory. Each field of the struct is placed after the previous one though the fields may not be contiguous due to alignment requirements. This can be observed in the gdb debugger. (In this example, l2 is a struct book):

```
(gdb) print l2
$1 = {title = "Vow", '\000' <repeats 44 times>,
  author = "Kim Carpenter", '\000' <repeats 34 times>,
  price = 7.4500000000000002, year = 2012, next = 0x601060 <l1>}

(gdb) print &l2
$2 = (book *) 0x6010e0 <l2>

(gdb) print &l2.title
$3 = (char (*)[48]) 0x6010e0 <l2>

(gdb) print &l2.author
$4 = (char (*)[48]) 0x601110 <l2+48>

(gdb) print &l2.price
$5 = (double *) 0x601140 <l2+96>

(gdb) print &l2.year
$6 = (unsigned int *) 0x601148 <l2+104>

(gdb) print &l2.next
$7 = (struct book **) 0x601150 <l2+112>
```

Using this information we can define *offsets* for each field of the struct:

```
; Offsets for fields in struct book.
;
%define TITLE_OFFSET 0
%define AUTHOR_OFFSET 48
%define PRICE_OFFSET 96
%define YEAR_OFFSET 104
%define NEXT_OFFSET 112
```

For example, if the RSI register holds the address of l2, then [RSI+YEAR_OFFSET] can be used to reference the year field of the struct. Similarly,[RSI+RCX+SUBJECT_OFFSET] can be used to access the *i*-th character of the subject string where the value of *i* is stored in RCX.

The last field of struct book is a pointer --- i.e., an address -- of another struct book. This field allows us to connect the nodes into a linked list. In the debugger session above, we can print out the address of another struct book and notice that it is the one referenced in the next field of l2.

(gdb) print l2.next
$1 = (struct book *) 0x601060 <l1>

(gdb) print &l1
$8 = (book *) 0x601060 <l1>

Assignment

Your assignment is to write an assembly language subroutine named authorsForPrice. The subroutine will be able to access a (double precision floating point) price in $, loaded into ST0 of the x87 Floating Point Unit's stack. The subroutine should traverse a linked list of struct book's and prints out all the author names for the e-books that have prices greater than (>) the loaded price in ST0. Each author name should be followed by a linefeed. You must make good use of indexed addressing modes.

CMSC 313 Project 4
Fall '18 Due: 10/16/2018 @ 11:59pm

The class blackboard has been updated with a `Project 4` directory. This directory already contains the skeleton of your project along with a Makefile that can build and test your project and tell you if it seems to be operating correctly.

Inside the skeleton project you will find the following important files:

- `library.o`: Defines a linked list of books that you can use to test your code. It exports the head of the linked list as a struct book pointer named "library"
- `driver.asm`: Invokes your authorsForDate function with several year parameters to verify that it is operating correctly
- `proj4.asm`: A starting point for creating your project
- `Makefile`: Describes to the "make" command how to build the C and assembly code described above and link everything together into the `proj4` executable
- `expected.txt`: Contains the output that a correct program should print when stimulated using the provided `driver.asm`

To build your project (from within the provided project4 directory):

```
make
```

To test your project (from within the provided project4 directory):

```
make test
```

The records in the library.o file contain the information for these e-books:

```
Breaking Point", "Pamela Clare", 6.85, 2011
Vow", "Kim Carpenter", 7.45, 2012
1491", "Charles C. Mann", 2.98, 2006
Three Weeks with My Brother", "Nicholas Sparks", 5.67, 2004
The 10 Habits of Happy Mothers", "Meg Meeker, M.D.", 8.56, 2011
We Need to Talk About Kevin", "Lionel Shriver", 4.65, 2011
The Prague Cemetery", "Umberto Eco", 7.34, 2011
Black Dahlia & White Rose", "Joyce Carol Oates", 8.25, 2012
Glad Tidings", "Debbie Macomber", 7.23, 2012
A Summer Affair", "Elin Hilderbrand", 4.25, 2008
Dead until Dark", "Charlaine Harris", 7.25, 2001
Between the Dark and the Daylight", "Richard Marsh", 8.56, 2012
Good Girls Don't", "Victoria Dahl", 8.24, 2011
Inheritance", "Christopher Paolini", 7.46, 2011
Best Staged Plans", "Claire Cook", 5.12, 2011
The Gap Year", "Sarah Bird", 6.35, 2011
The Affair Next Door", "Anna Katherine Green", 4.76, 2012
Crossed", "Ally Condie", 7.32, 2011
```

```
Eclipse", "Stephenie Meyer", 6.25, 2007
New Moon", "Stephenie Meyer", 5.25, 2007
The Third Gate", "Lincoln Child", 6.15, 2012
1225 Christmas Tree Lane", "Debbie Macomber", 7.24, 2011
Murder on Astor Place", "Victoria Thompson", 8.25, 2009
Heaven is for Real", "Todd Burpo", 6.78, 2011
15 Seconds", "Andrew Gross", 8.24, 2012
Harry Potter and the Chamber of Secrets", "J.K. Rowling", 8.99, 2012
The Handmaids Tale", "Margaret Atwood", 9.99, 1986
The Affair", "Lee Child", 8.23, 2011
The Girl With the Dragon Tattoo", "Stieg Larsson", 7.91, 2011
Explosive Eighteen", "Janet Evanovich", 5.24, 2011
The Help", "Kathryn Stockett", 3.47, 2009
```

The `library.o` file contains a pointer named library that points to the first node in the linked list. Thus, one of your first instructions should be:

```
mov rsi, [library]
```

When you are ready to look at the next node of the linked list, you can just say:

```
mov rsi, [esi+NEXT_OFFSET]
```

Implementation Notes

1. You need to declare library as an external label in your assembly language program: extern library. This is already done for you in the skeleton project.
2. This linked list does not have a "dummy" header node. The library pointer references the first node directly.
3. The next field of the last node in the linked list holds the value 0. That is how you will know that you have reached the end of the linked list.
4. When you print a string field of struct book (i.e., title, author and subject), you should NOT copy these strings to a new memory location, since these strings are already in memory. You should tell the WRITE system call where these strings reside.
5. Note that the strings in struct book are C-style NULL terminated strings. You will need to find out how long each string is before you make the system call to WRITE. (You can also use indexed addressing, here.) Example code is provided in the skeleton project. Feel free to use it as long as you add comments to prove that you know how it works.
6. Your project submission will be graded with a different data file. So, you should make sure that your code works when it's given a different linked list.
7. You may find the LEA instruction useful. (LEA = Load Effective Address.) The LEA instruction stores what would have been the address in the source operand into the destination operand. For example,

   ```
   lea rdi, [RSI+SUBJECT_OFFSET]
   ```

will compute RSI + SUBJECT_OFFSET and store the result in RDI.

8. You may also need to check the status of x87 FPU after and FCOMP using the FSTSW instruction.

```
fstsw AX            ; loads the current status of the FPU onto AX
```

Testing your program

Test your code by typing:

$ make test

The output of make test will simply be "Program output matches expected" when your program passes all of the public tests.

If your program fails the public tests then make test shows the lines that were expected (prefixed by <) and the lines that were seen (prefixed by >) along with the test "Program output does not match expected"

Turning in your program

Use the UNIX submit command on the GL system to turn in your project. You should submit 1 file: Your assembly language program named `proj4.asm`. You do not need to submit the library file, the driver, or a typescript file.

The UNIX command to do this should look exactly like:

```
submit CMSC313_Deepak proj4.asm driver.asm library.o Makefile readme.txt
```