DEPARTMENT OF COMPUTER SCIENCE
COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

CSCI 341: Computer Organization
HW 6: Recursion

When writing code, comments are important! Note the rubric specifies that about half your lines should be commented - this is not atypical for assembly.

We strongly recommend you write the pseudocode first (it is a required element in the submitted README), and if needed, rewrite it using only the constructs if-goto and goto, to make the leap from abstraction to RISC-V smaller. Have your pseudocode handy if you come to office hours for assistance with this.

Search "recursion" on Google. Try it, because I guarantee no matter how correctly you spell it, Google will always ask "Did you mean: recursion". This is a small humorous example of recursion, where if you click on the "Did you mean…" link you will get right back to where you were, with the exact same "Did you mean…" link.

In this program you will be implementing a recursive binary search procedure, printing out the summation of each sub array at each level of recursion (i.e., each time the procedure begins). Why print out the summation value at each level of recursion? Three reasons:

1. You will get practice calling procedures
2. You will get to use the sum procedure that you wrote in the previous programming assignment
3. It makes it so we know if you are actually implementing binary search vs just a simple linear search.

This program will be written in a similar manner to the previous one, with the binarySearch procedure declared `.globl`. This assignment's grader script (provided in Canvas) will be used when grading to call your procedure. It is different from last week's grading script! When prompted to add a new number to the array, you should make sure the array is sorted.

To test your code, two files need to be in the same directory and open in RARS: this file with your binarySearch procedure and your sum procedure copied from last week's submission, and the grading script file, *with no other files open*. Under the settings menu in RARS check the "Assemble all files currently open" and "Initialize Program Counter to global main if defined" options. Your files should *not* have a main: label in them (it's in the grader script). You are welcome to copy any of the macros provided in example_macros.s into your file that you wish to use in your code (see 1/25 handouts on the Section A schedule page in Canvas).

Your binarySearch procedure can assume the array is sorted. The procedure must match the following signature:

1. Name: binarySearch
2. Arguments:
   a2: Address of the array
   a3: start index
   a4: end index
   a5: value to find
3. Return Value:
   a0: index of value

For a description of the sum procedure, see last week's homework. Note that to sum only a portion of an array, you would provide the address of the first entry to print and the number of entries to print including that one.

An example run through of what the program should do when given an array of {1,2,3,4,5,6,7,8,9,10}, start index of 0, end index of 9, and a value of 1 is shown below. The program's output is also shown. Note that when computing the summation at each step the start and end indexes are both inclusive. The general formula for computing the mid index is as follows:

$$mid = \frac{endIndex+startIndex}{2}$$

RISC-V does include a `div` instruction; note that it does not error out if your divisor is 0, and it rounds toward 0.

Array	Start Index	End Index	Mid Index	Summation
{1,2,3,4,5,6,7,8,9,10}	(Given) 0	(Given) 9	(9+0)/2=4	1+2+...+10=55
{1,2,3,4,5,6,7,8,9,10}	0	4-1=3	(3+0)/2=1	1+2+3+4=10
{1,2,3,4,5,6,7,8,9,10}	0	1-1=0	(0+0)/2=0	1=1

```
Enter a number to add to the array: 1
Enter a number to add to the array: 2
Enter a number to add to the array: 3
Enter a number to add to the array: 4
Enter a number to add to the array: 5
Enter a number to add to the array: 6
Enter a number to add to the array: 7
Enter a number to add to the array: 8
```

```
Enter a number to add to the array: 9
Enter a number to add to the array: 10
Enter a number to add to the array: 0
What value are you looking for? 1
Summation: 55
Summation: 10
Summation: 1
Index: 0
Value: 1

-- program is finished running (0) --
```

Note how when a "0" is entered the grading script stops accepting values. Also, note how the array length is variable. Your binarySearch procedure is expected to work with arrays of any size. As always, assume a well-behaved user (values entered are in order, though not necessarily sequential, and the value to find is always in the list).

A second test case given the array {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and value 9:

Array	**Start Index**	**End Index**	**Mid Index**	**Summation**
{1,2,3,4,5,6,7,8,9,10}	(Given) 0	(Given) 9	(9+0)/2=4	1+2+...+10=55
{1,2,3,4,5,6,7,8,9,10}	4+1=5	9	(9+5)/2=7	6+7+8+9+10=40
{1,2,3,4,5,6,7,8,9,10}	7+1=8	9	(9+8)/2=8 (finds value at mid-index and returns)	9+10=19

What to turn in:

1. A zipped file with the file name lastnameFirstname.zip (.zip is the file extension) containing:
    a. Your README, with the file name lastnameFirstname_README.txt (.txt is the file extension, this should be a plain text file).
    b. Your code, with the file name lastnameFirstname_ASSEMBLY.s **Do not submit the file you downloaded to test your code.** (.s is the file extension)

DEPARTMENT OF COMPUTER SCIENCE
COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

In your README, include the following elements:

1. Include YOUR NAME and the names of all people who helped/collaborated as per the syllabus and CS@Mines collaboration policy. [This is an individual assignment]
2. Pseudocode (in C/Java/Python/etc) for your implementation [yes it should also be recursive].
3. Describe the challenges you encountered and how you surmounted them.
4. What did you like about the assignment?
5. How long did you spend working on this assignment?
6. A description of any features you added for extra credit (if any)

Extra Credit Options (See separate canvas assignment for more details):

1. Create a separate procedure that first sorts the given array then calls binary search
2. Create a separate procedure that first checks if the given array is sorted. If it is then call binary search and if not return -1
3. Create a separate procedure that implements a different search algorithm

Section	Points
Procedures Implemented Properly: Caller: *jal ra, LABEL* Callee: *jalr zero, ra,* 0 binarySearch implemented recursively This is a big part of this grade: DOUBLE CHECK IT IS DONE CORRECTLY	10 points
Public Test Cases Pass	2 points each, 4 points total
Private Test Cases Pass	2 points each, 20 points total
Adequate comment coverage (50%)	5 pts
No pseudo instructions other than la to load addresses (hint: la should not be needed for this assignment)	6 pts, -1 for each up to -6 pts
README has required elements and effort is demonstrated	5 pts
E.C. 5 points max Up to the graders discretion	+5 points
Total	50 pts (55 with e.c.)

DEPARTMENT OF
COMPUTER SCIENCE

COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT