Eric Song
I pledge my honor that I have abided by the Stevens Honor System.
November 3rd, 2021

Lab 9 Writeup

Like project1, just in case, a copy of my code is attached at the end of this document.

My code is broken into two main parts, separated by a visual comment filled with many slashes (/). The first part is directly taken from my Project 1, selection sort, with the print loop removed. The second part is my implementation of binary search.

There are four edge cases needed to be tested. In Part (0), I'll attach a screenshot of my code actually running normally. The array in .data is [5, 3, 4, -13, 1]. Selection Sort (from project1) was run on this array to sort it. The sorted array becomes [-13, 1, 3, 4, 5]. For reference, the four parts to the test cases are :

(1) the query is not in the array,
(2) the query is the first element of the array,
(3) the query is the last element of the array,
(4) the query is some position in the array.

Part 0:

```
hyogwah@hyo: ~/assembly/binarysearch
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
-13
Found in the array: -13
At index: 0
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
1
Found in the array: 1
At index: 1
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
3
Found in the array: 3
At index: 2
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
4
Found in the array: 4
At index: 3
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
5
Found in the array: 5
At index: 4
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
6
Not found in the array: 6
hyogwah@hyo:~/assembly/binarysearch$ qemu-aarch64 a.out
-100
Not found in the array: -100
hyogwah@hyo:~/assembly/binarysearch$
```

Part 1) The query is not in the array

```
CMP X19, X22
B.LT ifAmlessthanT // if (X27 < X22)
CMP X19, X22
B.GT ifAmgreaterthanT
B EXITT //return m at exit1


ifAmlessthanT: // L = m+1
ADD X27, X25, 1
MOV X23, X27

CMP X23, X24 // while L ≤ R do
B.GT unsuccessful
B L1

ifAmgreaterthanT: // R = m-1
SUB X27, X25, 1
MOV X24, X27

CMP X23, X24 // while L ≤ R do
B.GT unsuccessful
B L1
```

I had a series of conditional statements that compared the values of X19 and X22, which are A[m] and T respectively, with T being the target, and m being the midpoint of the left + right / 2 number.

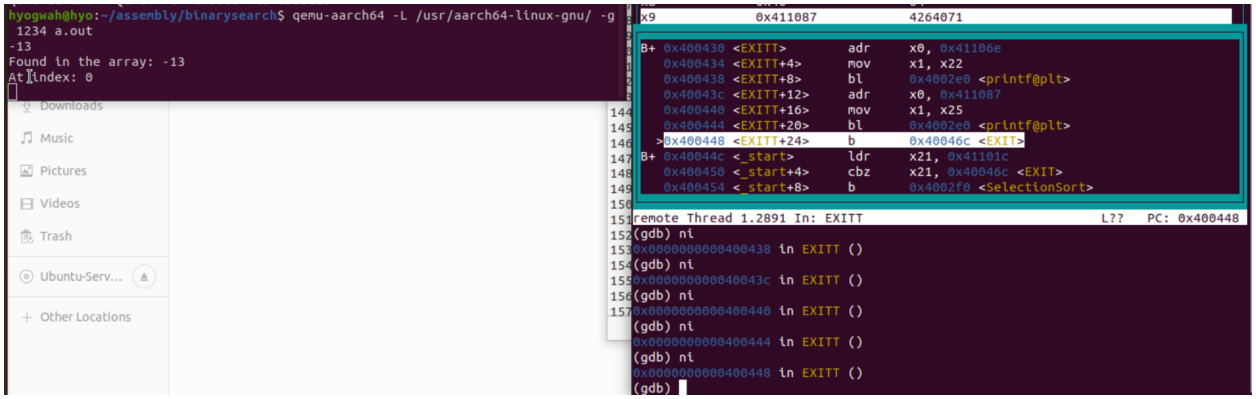
```
hyogwph@hyo:~/assembly/binarysearch$ qemu-aarch64 -L /usr/aarch64-linux-gnu/ -g
 1234 a.out
15
Not found in the array: 15
```

```
hyogwah@hyo: ~/assembly/binarysearch
Register group: general
x0             0x411051            4264017
x1             0x0                 0
x2             0x0                 0
x3             0xffffffd0          4294967248
x4             0x5                 5
x5             0x40007ffc22        274886294562
x6             0x21a               538
x7             0xf                 15
x8             0x400095c1a8        274887721384
x9             0x5                 5

B+ 0x400420 <unsuccessful>          adr     x0, 0x411051
  >0x400424 <unsuccessful+4>        mov     x1, x22
   0x400428 <unsuccessful+8>        bl      0x4002e0 <printf@plt>
   0x40042c <unsuccessful+12>       b       0x40046c <EXIT>
   0x400430 <EXITT>                 adr     x0, 0x41106e
   0x400434 <EXITT+4>               mov     x1, x22
   0x400438 <EXITT+8>               bl      0x4002e0 <printf@plt>
   0x40043c <EXITT+12>              adr     x0, 0x411087
   0x400440 <EXITT+16>              mov     x1, x25
   0x400444 <EXITT+20>              bl      0x4002e0 <printf@plt>
remote Thread 1.2884 In: unsuccessful                    L??   PC: 0x400424

Breakpoint 3, 0x0000000000400420 in unsuccessful ()
(gdb) ni
0x0000000000400424 in unsuccessful ()
(gdb) x/40db $x20
0x411024:       -13     -1      -1      -1      -1      -1      -1      -1
0x41102c:       1       0       0       0       0       0       0       0
0x411034:       3       0       0       0       0       0       0       0
0x41103c:       4       0       0       0       0       0       0       0
0x411044:       5       0       0       0       0       0       0       0
(gdb)
```

You can clearly see in gdb that the array (stored in X20) does not contain the input value of 15. The program prints out Not found, as expected.

Part 2) the query is the first element of the array:

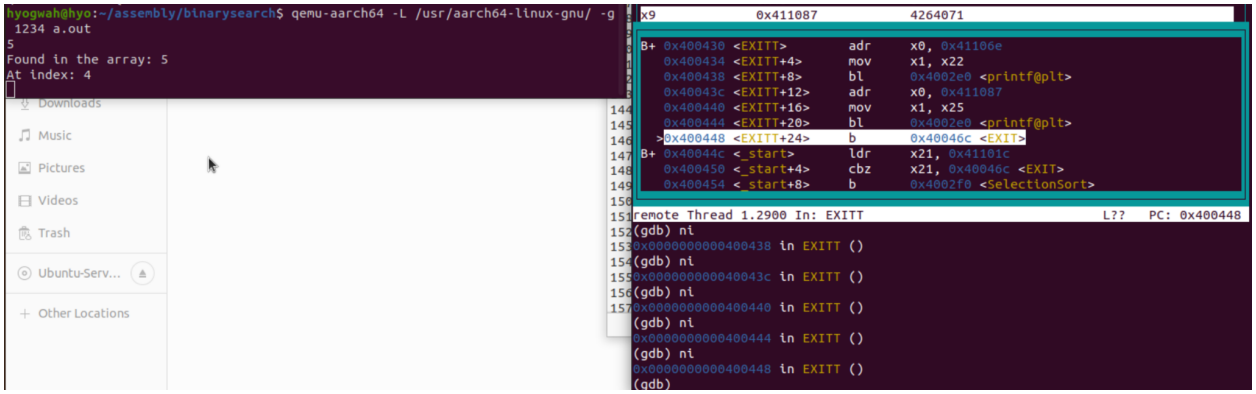
When the query is the first element of the array, it should be -13.



You can see that after I input -13 in the scanf, the function loops as normal in the while loop, and then branches to EXITT, where then I printf the number found, along with a second printf where I print the index of the input I found. See comments in code for more detail.

Part 3) the query is the last element of the array

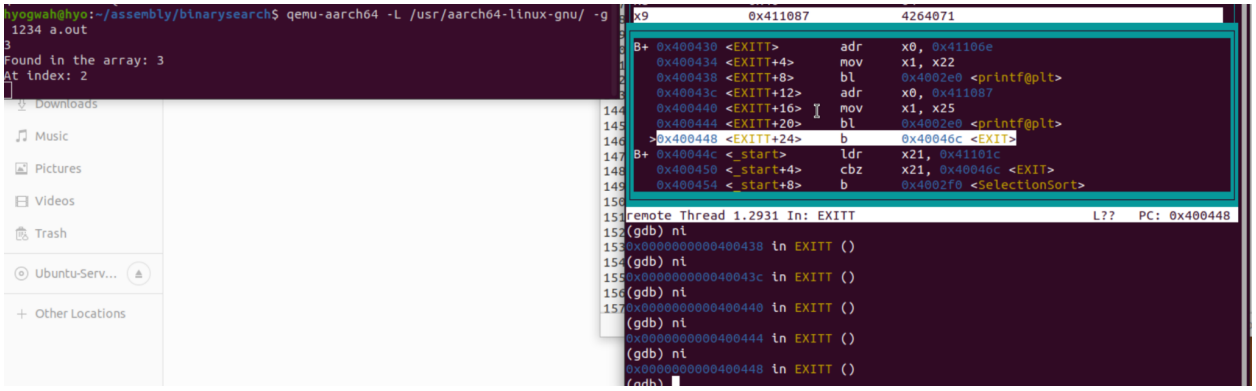
The last element in the demo array stored in .data is 5 (remember [-13, 1, 3, 4, 5] after selection sort). The index should be 4, and the result should tell us that the element has been found. With that being said, gdb should show us that we go to EXITT, which is where the function goes if the element has been found.



You can see that 5 has been found, properly branching to EXITT, with the proper index of 4. Yay. Hurray.

Part 4) The query is in some position of the array

I think part 2 and 3 should be sufficient, but I will show a random number, let's say 3, which is neither the edge cases (first or last element). Same exact process.



You can see that in my array [-13, 1, 3, 4, 5], 3 is index 2, and it is indeed found.

```
/*     0  1  2  3  4
A = [-13, 1, 3, 4, 5]
function binary_search(A, 5, 1)
L = 0
R = 4
while (1 <= 1)
   m = 1
   if A[1] < 1
       L = 1
   if  3 > 1
       R = 2-1 =1
   return 1:
function binary_search(A, n, T) is
   L := 0
   R := n - 1
   while L ≤ R do
       m := floor((L + R) / 2)
       if A[m] < T then
           L := m + 1
       else if A[m] > T then
           R := m - 1
       else:
           return m
   return unsuccessful

*/

// Eric Song
// I pledge my honor that I have abided by the Stevens Honor System
.text
.global _start
.extern printf
.extern scanf

/*
void selection_sort(int array[], const int length) {
   for(int i = 0; i < length - 1; i++) {
       int min_j = i;
```

```
        for(int j = i + 1; j < length; j++) {
            if(array[j] < array[min_j]) {
                min_j = j;
            }
        }
        if(min_j != i) {
            swap(array, i, min_j);
        }
    }
}
*/

/*
X20 base address of array
X21 size
X22 i
X23 min_j
X24 j
X25 value of array[j]
X26 value of array[min_j]
X27 length-1
X28 i+1
X29 print index
*/

SelectionSort:
   ADR X20, arr //loads base address of array
   LDR X21, size
   MOV X27, X21
   SUB X27, X27, 1 // length - 1
   // for(int i = 0; i < length - 1; i++)
   MOV X22, 0
OUTER:
   MOV X23, X22 //int min_j = i;

   MOV X28, X22 // x28 = i
   ADD X28, X28, 1 // x28 = i+1
   MOV X24, X28
INNER:
   LSL X1, X24, 3 // this offset of j * 8
   ADD X9, X20, X1 // base + offset of array
   LDUR X10, [X9, 0] // value of A[j] inside X10
```

```
   LSL X2, X23, 3 // this offset of min_j * 8
   ADD X11, X20, X2 // base + offset of array
   LDUR X12, [X11, 0] // value of A[min_j] inside X12

   CMP X12, X10
   B.GT IFMINJGREATERTHANJ // if X12 > X10
   ADD X24, X24, 1
   CMP X24, X21
   B.NE INNER
   B AFTERINNER

IFMINJGREATERTHANJ:
   MOV X23, X24
   ADD X24, X24, 1
   CMP X24, X21
   B.NE INNER
   B AFTERINNER

// end of inner loop
AFTERINNER:
   CMP X23, X22
   B.NE pleaseSwap
   B DONE

pleaseSwap:
   BL SWAP

DONE:
   ADD X22, X22, 1
   CMP X22, X27 //compares i and (length-1)
   B.NE OUTER // if i != length-1
   B startofsearch

SWAP:
   SUB SP, SP, 8
   STUR LR, [SP, 0]

   LSL X13, X22, 3
   ADD X14, X20, X13
   LDUR X15, [X14, 0]

   LSL X16, X23, 3
```

```
   ADD X17, X20, X16
   LDUR X18, [X17, 0]
   MOV X3, X15
   MOV X15, X18
   MOV X18, X3

   STUR X15, [X14, 0]
   STUR X18, [X17, 0]

   LDUR LR, [SP, 0] //loading return address to start
   ADD SP, SP, 8 //adding it back (pop) == deallocating
   BR LR // or X30
////////////////////////////////////////////////////////////////////////////////

binaryserach:
   LDR X28, size
   MOV X23, 0       // L = 0
   SUB X27, X28, 1   // X27 = n - 1
   MOV X24, X27     // R := n - 1

L1: // while L ≤ R do

   ADD X27, X23, X24 // L + R
   LSR X27, X27, 1       // m := floor((L + R) / 2)
   MOV X25, X27     // X25(m) = floor(L+R) /2)

   MOV X29, X25     // X29 = m
   LSL X29, X29, 3       // m * 8 for index
   ADD X15, X20, X29 // address + index into X15
   LDUR X19, [X15, 0] // X27 = A[m]

   CMP X19, X22
   B.LT ifAmlessthanT // if (X27 < X22)
   CMP X19, X22
   B.GT ifAmgreaterthanT
   B EXITT //return m at exit1


   ifAmlessthanT: // L = m+1
   ADD X27, X25, 1
   MOV X23, X27
```

```
   CMP X23, X24 // while L ≤ R do
   B.GT unsuccessful
   B L1

   ifAmgreaterthanT: // R = m-1
   SUB X27, X25, 1
   MOV X24, X27

   CMP X23, X24 // while L ≤ R do
   B.GT unsuccessful
   B L1

unsuccessful:
   ADR X0, bad
   MOV X1, X22
   BL printf
   B EXIT

EXITT: //success
   ADR X0, good
   MOV X1, X22
   BL printf

   ADR X0, index
   MOV X1, X25
   BL printf

   B EXIT

_start:
   LDR X21, size
   CBZ X21, EXIT
   B SelectionSort

startofsearch:
   ADR X0, target
   ADR X1, input
   BL scanf
   LDR X22, input
   B binaryserach
   // exit call
```

```
EXIT:
   MOV X0, 0
   MOV W8, 93
   SVC 0
/*
X20 base address of array
X21 size (n)
X22 target (T)
X23 L
X24 R
X25 m
X26 counter for while loop
X27 temp
X28 temp (size)
X29 index i
*/

.data
target: .ascii "%ld\0"
input: .quad 0
size: .quad 5
arr: .quad 5, 3, 4, -13, 1
str: .ascii "%ld\n\0" // quad, use %ld to print
bad: .ascii "Not found in the array: %ld\n\0" // quad, use %ld to print
good: .ascii "Found in the array: %ld\n\0" // quad, use %ld to print
index: .ascii "At index: %ld\n\0" // quad, use %ld to print
.end
```