

## ECE 331 Lab 1: Implementing a sorting algorithm for Primes and Composites in Arm Assembly

Code:

(All of it is on a GitHub repo, it is private so no one uses my code while the project is live)

```
.globl isPrimeAssembly

isPrimeAssembly: // Same as primeIterator()
    // Your code for iterating through arrays here
    // The Base addresses of arrays in X0 - X2, length in X3s
    // X0 has the base address of Original array
    // X1 has the base address of Prime array
    // X2 has the base address of Composite array
    // X3 has the length of the Original array
    nop    //first line of execution when called by main.c
    mov x20, x30 // Save the return register for later
    mov x4, #0 // Initialize as x4=index=0

iterate:
    ldr x19, [x0, x4, lsl #3] // Load value from original array (x0), using lsl to use a
multiple of 8
    // x19 is going to be our n value
    bl isPrime // Call isPrime sub-function, branch to label

    cbz x6, storeComposite // If the returnZero label is used and X6 is 0
    str x19, [x1, x4, LSL #3] // Else store x19 into the Prime array at the current index
    b nextIndex // Branch to calculate next index

storeComposite:
    str x19, [x2, x4, lsl #3] // Store x19 into the Composite array at the current index
    b nextIndex // Branch to calculate next index

nextIndex:
    add x4, x4, #1 // Index=x4+=1, increase index by 1
    cmp x3, x4 // Compare length of original array to altered x4 index (Index that is not a
multiple of 8)
    b.ne iterate // Continue iterating until all array values are accessed
    mov x30, x20 // Reload value of branch to x30
    ret // Return to C code

isPrime: // Same as isPrime()
    // Your code for detecting a prime number here
    mov x9, #2 // Set x9=i=2
    lsr x10, x19, #1 // Set x10=n/2 using logical shift right by 1

loopOne:
    cmp x9, x10 // Compare i with n/2 and raise flags
    bgt returnOne // If i > n/2, bgt with register equal to 1

    udiv x11, x19, x9 // This line computes x11 = quotient = n / i
    msub x12, x11, x9, x19 // This line computes x12 = x19 - x11 * x9 = n - q*i
    cbz x12, returnZero // If x12 reaches 0, go to returnZero label
```

```

add x9, x9, #1 // x9=i+=1, increase i by 1
b loopOne // Go back to loopOne label to keep calculating

```

returnOne:

```

mov x6, #1 // Set x6 to 1 for storing into array
br x30 // Branch to last link to store a Prime

```

returnZero:

```

mov x6, #0 // Set x6 to 0 for storing into array
br x30 // Branch to last link to store a Composite

```

Images:

(Some commands for me were different as I was running this on my native Arch linux install instead of the provided Ubuntu Virtual machine)

This arrayA stored at x0 as seen using x/16gd \$x0 in gdb:

The screenshot shows a GDB session with the following components:

- Register window:** Displays the state of all 32 registers. Register x0 contains the address 0x7f06dd8821d8.
- Disassembly window:** Shows the assembly code for the function `isPrimeAssembly`. The current instruction is `mov x20, x30` at address 0x4009f4.
- Memory dump:** A command `x/16gd $x0` has been executed, displaying the contents of the array stored at x0. The array contains 16 integers: 16, 40, 39, 10, 18, 83, 5, 11, 16, 40, 39, 10, 18, 83, 5, 11.

This arrayPrime stored at x1 as seen using x/16gd \$x1 in gdb:

GROUP 1

C main.c

GROUP 2

isPrimeAssembly.s

ECE331 PROJECT 1

ECE331\_Project1\_v3.pdf

fact

fact.o

fact.s

isPrime

isPrimeAssembly.s

main.c

qemu\_isPrime\_20241104-223546\_38930\_

read.me

Register group: general

x0	0x7f06dd8821d8	139667463217624	x1	0x7f06dd882158	139667463217496
x2	0x7f06dd8820d8	139667463217368	x3	0x10	16
x4	0x7f06dcff0e80	139667454234240	x5	0x8130989426fb24d3	-9137635881858161453
x6	0x7f06dcfe1e60	139667454172768	x7	0x7f06dd07fb00	139667454819072
x8	0xd7	215	x9	0x18	24
x10	0x0	0	x11	0x0	0
x12	0x7f06dd082350	139667454829392	x13	0x2e	46
x14	0x3d8f538	64550200	x15	0x7f06dcff1cc0	139667454237888
x16	0x7f06dcfd68	139667454164328	x17	0x7f06dd0536d0	139667454637776
x18	0x3	3	x19	0x7f06dd8823e8	139667463218152
x20	0x1	1	x21	0x41fd0	4324848
x22	0x4005f4	4195828	x23	0x7f06dd8823f8	139667463218168
x24	0x7f06dd080b68	139667454823272	x25	0x0	0
x26	0x7f06dd061000	139667454824448	x27	0x41fd0	4324848
x28	0x0	0	x29	0x7f06dd8820c0	139667463217344
x30	0x4008dc	4196572	sp	0x7f06dd8820c0	0x7f06dd8820c0

B+ 0x4009f0 <isPrimeAssembly>

0x4009f4 <isPrimeAssembly+4>

0x4009f8 <isPrimeAssembly+8>

0x4009fc <iterate>

0x400a00 <iterate+4>

0x400a04 <iterate+8>

0x400a08 <iterate+12>

0x400a0c <iterate+16>

0x400a10 <storeComposite>

0x400a14 <storeComposite+4>

0x400a18 <nextIndex>

0x400a1c <nextIndex+4>

0x400a20 <nextIndex+8>

0x400a24 <nextIndex+12>

0x400a28 <nextIndex+16>

0x400a2c <isPrime>

remote Thread 21285.21285 (regs) In: isPrimeAssembly

L11 PC: 0x4009f4

0x7f06dd8821d8: 7

0x7f06dd8821e8: 23

0x7f06dd8821f8: 11

0x7f06dd882208: 37

0x7f06dd882218: 2

0x7f06dd882228: 44

0x7f06dd882238: 87

0x7f06dd882248: 6

(gdb) x/16gd \$x1

0x7f06dd882158: 0

0x7f06dd882168: 0

0x7f06dd882178: 0

0x7f06dd882188: 0

0x7f06dd882198: 0

0x7f06dd8821a8: 0

0x7f06dd8821b8: 0

0x7f06dd8821c8: 0

(gdb) []

This arrayComposite stored at x2 as seen using x/16gd \$x2 in gdb:

GROUP 1

C main.c

GROUP 2

isPrimeAssembly.s

ECE331 PROJECT 1

ECE331\_Project1\_v3.pdf

fact

fact.o

fact.s

isPrime

isPrimeAssembly.s

main.c

qemu\_isPrime\_20241104-223546\_38930\_

read.me

Register group: general

x0	0x7f06dd8821d8	139667463217624	x1	0x7f06dd882158	139667463217496
x2	0x7f06dd8820d8	139667463217368	x3	0x10	16
x4	0x7f06dcff0e80	139667454234240	x5	0x8130989426fb24d3	-9137635881858161453
x6	0x7f06dcfe1e60	139667454172768	x7	0x7f06dd07fb00	139667454819072
x8	0xd7	215	x9	0x18	24
x10	0x0	0	x11	0x0	0
x12	0x7f06dd082350	139667454829392	x13	0x2e	46
x14	0x3d8f538	64550200	x15	0x7f06dcff1cc0	139667454237888
x16	0x7f06dcfd68	139667454164328	x17	0x7f06dd0536d0	139667454637776
x18	0x3	3	x19	0x7f06dd8823e8	139667463218152
x20	0x1	1	x21	0x41fd0	4324848
x22	0x4005f4	4195828	x23	0x7f06dd8823f8	139667463218168
x24	0x7f06dd080b68	139667454823272	x25	0x0	0
x26	0x7f06dd061000	139667454824448	x27	0x41fd0	4324848
x28	0x0	0	x29	0x7f06dd8820c0	139667463217344
x30	0x4008dc	4196572	sp	0x7f06dd8820c0	0x7f06dd8820c0

B+ 0x4009f0 <isPrimeAssembly>

0x4009f4 <isPrimeAssembly+4>

0x4009f8 <isPrimeAssembly+8>

0x4009fc <iterate>

0x400a00 <iterate+4>

0x400a04 <iterate+8>

0x400a08 <iterate+12>

0x400a0c <iterate+16>

0x400a10 <storeComposite>

0x400a14 <storeComposite+4>

0x400a18 <nextIndex>

0x400a1c <nextIndex+4>

0x400a20 <nextIndex+8>

0x400a24 <nextIndex+12>

0x400a28 <nextIndex+16>

0x400a2c <isPrime>

remote Thread 21285.21285 (regs) In: isPrimeAssembly

L11 PC: 0x4009f4

0x7f06dd882158: 0

0x7f06dd882168: 0

0x7f06dd882178: 0

0x7f06dd882188: 0

0x7f06dd882198: 0

0x7f06dd8821a8: 0

0x7f06dd8821b8: 0

0x7f06dd8821c8: 0

(gdb) x/16gd \$x2

0x7f06dd8820d8: 0

0x7f06dd8820e8: 0

0x7f06dd8820f8: 0

0x7f06dd882108: 0

0x7f06dd882118: 0

0x7f06dd882128: 0

0x7f06dd882138: 0

0x7f06dd882148: 0

(gdb) []

After stepping through the code:

This arrayA stored at x0 as seen using x/16gd \$x0 in gdb (Should not have changed):

Register group: general

x2	0x7f06dd8820d8	139667463217368	x3	0x10	16
x4	0x10	16	x5	0x8130989426fb24d3	-9137635881858161453
x6	0x1	1	x7	0x7f06dd07fb00	139667454819072
x8	0xd7	215	x9	0x6	6
x10	0x5	5	x11	0x2	2
x12	0x1	1	x13	0x2e	46
x14	0x3d8f538	64550200	x15	0x7f06dcff1cc0	139667454237888
x16	0x7f06dcfd68	139667454164328	x17	0x7f06dd0536d0	139667454637776
x18	0x3	3	x19	0xb	11
x20	0x4008dc	4196572	x21	0x41fd0	4324848
x22	0x4005f4	4195828	x23	0x7f06dd8823f8	139667463218168
x24	0x7f06dd080b68	139667454823272	x25	0x0	0
x26	0x7f06dd081000	139667454824448	x27	0x41fd0	4324848
x28	0x0	0	x29	0x7f06dd8820c0	139667463217344
x30	0x400a04	4196868	sp	0x7f06dd8820c0	0x7f06dd8820c0
pc	0x400a1c	0x400a1c <nextIndex+4>	cpsr	0x20000000	536870912

```

0x4009fc <iterate> ldr x19, [x0, x4, lsl #3]
0x400a00 <iterate+4> bl 0x400a2c <isPrime>
0x400a04 <iterate+8> cbz x6, 0x400a10 <storeComposite>
0x400a08 <iterate+12> str x19, [x1, x4, lsl #3]
0x400a0c <iterate+16> b 0x400a18 <nextIndex>
0x400a10 <storeComposite> str x19, [x2, x4, lsl #3]
0x400a14 <storeComposite+4> b 0x400a18 <nextIndex>
0x400a18 <nextIndex> add x4, x4, #0x1
0x400a1c <nextIndex+4> cmp x3, x4
0x400a20 <nextIndex+8> b.ne 0x4009fc <iterate> // b.any
0x400a24 <nextIndex+12> mov x30, x20
0x400a28 <nextIndex+16> ret
0x400a2c <isPrime> mov x9, #0x2 // #2
0x400a30 <isPrime+4> lsr x10, x19, #1
0x400a34 <loopOne> cmp x9, x10
0x400a38 <loopOne+4> b.gt 0x400a50 <returnOne>

```

remote Thread 21285.21285 (regs) in: nextIndex L29 PC: 0x400a1c

```

iterate () at isPrimeAssembly.s:15
isPrime () at isPrimeAssembly.s:37
loopOne () at isPrimeAssembly.s:41
returnOne () at isPrimeAssembly.s:51
iterate () at isPrimeAssembly.s:19
nextIndex () at isPrimeAssembly.s:28
(gdb) x/16gd $x0
Value can't be converted to integer.
(gdb) x/16gd $x0
0x7f06dd8821d8: 7 16
0x7f06dd8821e8: 23 40
0x7f06dd8821f8: 11 39
0x7f06dd882208: 37 10
0x7f06dd882218: 2 18
0x7f06dd882228: 44 83
0x7f06dd882238: 87 5
0x7f06dd882248: 6 11
(gdb)

```

This arrayPrime stored at x1 as seen using x/16gd \$x1 in gdb:

Register group: general

x2	0x7f06dd8820d8	139667463217368	x3	0x10	16
x4	0x10	16	x5	0x8130989426fb24d3	-9137635881858161453
x6	0x1	1	x7	0x7f06dd07fb00	139667454819072
x8	0xd7	215	x9	0x6	6
x10	0x5	5	x11	0x2	2
x12	0x1	1	x13	0x2e	46
x14	0x3d8f538	64550200	x15	0x7f06dcff1cc0	139667454237888
x16	0x7f06dcfd68	139667454164328	x17	0x7f06dd0536d0	139667454637776
x18	0x3	3	x19	0xb	11
x20	0x4008dc	4196572	x21	0x41fd0	4324848
x22	0x4005f4	4195828	x23	0x7f06dd8823f8	139667463218168
x24	0x7f06dd080b68	139667454823272	x25	0x0	0
x26	0x7f06dd081000	139667454824448	x27	0x41fd0	4324848
x28	0x0	0	x29	0x7f06dd8820c0	139667463217344
x30	0x400a04	4196868	sp	0x7f06dd8820c0	0x7f06dd8820c0
pc	0x400a1c	0x400a1c <nextIndex+4>	cpsr	0x20000000	536870912

```

0x4009fc <iterate> ldr x19, [x0, x4, lsl #3]
0x400a00 <iterate+4> bl 0x400a2c <isPrime>
0x400a04 <iterate+8> cbz x6, 0x400a10 <storeComposite>
0x400a08 <iterate+12> str x19, [x1, x4, lsl #3]
0x400a0c <iterate+16> b 0x400a18 <nextIndex>
0x400a10 <storeComposite> str x19, [x2, x4, lsl #3]
0x400a14 <storeComposite+4> b 0x400a18 <nextIndex>
0x400a18 <nextIndex> add x4, x4, #0x1
0x400a1c <nextIndex+4> cmp x3, x4
0x400a20 <nextIndex+8> b.ne 0x4009fc <iterate> // b.any
0x400a24 <nextIndex+12> mov x30, x20
0x400a28 <nextIndex+16> ret
0x400a2c <isPrime> mov x9, #0x2 // #2
0x400a30 <isPrime+4> lsr x10, x19, #1
0x400a34 <loopOne> cmp x9, x10
0x400a38 <loopOne+4> b.gt 0x400a50 <returnOne>

```

remote Thread 21285.21285 (regs) in: nextIndex L29 PC: 0x400a1c

```

(gdb) x/16gd $x1
0x7f06dd8821d8: 7 16
0x7f06dd8821e8: 23 40
0x7f06dd8821f8: 11 39
0x7f06dd882208: 37 10
0x7f06dd882218: 2 18
0x7f06dd882228: 44 83
0x7f06dd882238: 87 5
0x7f06dd882248: 6 11
(gdb)

```

OPEN EDITORS

GROUP 1

main.c

GROUP 2

X == isPrimeAssembly.s

ECE331 PROJECT 1

ECE331\_Project1\_v3.pdf

fact

fact.o

fact.s

isPrime

isPrimeAssembly.s

main.c

qemu\_isPrime\_20241104-223546\_38930...

read.me

Register group: general

x2	0x7f06dd8820d8	139667463217368	x3	0x10	16
x4	0x10	16	x5	0x8130989426fb24d3	-9137635881858161453
x6	0x1	1	x7	0x7f06dd07fb00	139667454819072
x8	0xd7	215	x9	0x6	6
x10	0x5	5	x11	0x2	2
x12	0x1	1	x13	0x2e	46
x14	0x3d8f538	64550200	x15	0x7f06dcff1cc0	139667454237888
x16	0x7f06dcdfdf68	139667454164328	x17	0x7f06dd0536d0	139667454637776
x18	0x3	3	x19	0xb	11
x20	0x4008dc	4196572	x21	0x41fd0	4324848
x22	0x4005f4	4195828	x23	0x7f06dd8823f8	139667463218168
x24	0x7f06dd080b68	139667454823272	x25	0x0	0
x26	0x7f06dd081000	139667454824448	x27	0x41fd0	4324848
x28	0x0	0	x29	0x7f06dd8820c0	139667463217344
x30	0x400a04	4196868	sp	0x7f06dd8820c0	0x7f06dd8820c0
pc	0x400a1c	0x400a1c <nextIndex+4>	cpsr	0x20000000	536870912

```

0x4009fc <iterate>      ldr    x19, [x0, x4, lsl #3]
0x400a00 <iterate+4>    bl     0x400a2c <isPrime>
0x400a04 <iterate+8>    cbz    x6, 0x400a10 <storeComposite>
0x400a08 <iterate+12>   str    x19, [x1, x4, lsl #3]
0x400a0c <iterate+16>   b      0x400a18 <nextIndex>
0x400a10 <storeComposite> str    x19, [x2, x4, lsl #3]
0x400a14 <storeComposite+4> b      0x400a18 <nextIndex>
0x400a18 <nextIndex>    add     x4, x4, #0x1
0x400a1c <nextIndex+4>  cmp     x3, x4
0x400a20 <nextIndex+8>  b.ne   0x4009fc <iterate> // b.any
0x400a24 <nextIndex+12> mov     x30, x20
0x400a28 <nextIndex+16> ret
0x400a2c <isPrime>      mov     x9, #0x2 // #2
0x400a30 <isPrime+4>    lsr     x10, x19, #1
0x400a34 <loopOne>      cmp     x9, x10
0x400a38 <loopOne+4>    b.gt   0x400a50 <returnOne>

```

remote Thread 21285.21285 (regs) In: nextIndex

0x7f06dd8821d8: 7	16
0x7f06dd8821e8: 23	40
0x7f06dd8821f8: 11	39
0x7f06dd882208: 37	10
0x7f06dd882218: 2	18
0x7f06dd882228: 44	83
0x7f06dd882238: 87	5
0x7f06dd882248: 6	11
(gdb) x/16gd \$x1	
0x7f06dd882158: 7	0
0x7f06dd882168: 23	0
0x7f06dd882178: 11	0
0x7f06dd882188: 37	0
0x7f06dd882198: 2	0
0x7f06dd8821a8: 0	83
0x7f06dd8821b8: 0	5
0x7f06dd8821c8: 0	11

(gdb) █

Here's me running my code without a server, the C function is commented out just like the above gdb images!

The screenshot displays a development environment with three main sections:

- Left Pane (C Source Code):**

```

22 void primeIterator(unsigned long long a[], unsigned long long prime[], unsigned long long composite[]) {
23     return;
24 }
25
26 int main() {
27     // Initialize test and result arrays
28     unsigned long long arrayLength = 16;
29     unsigned long long a[] = {7, 16, 23, 40, 11, 39, 37, 10, 2, 18, 44, 8, 1, 1, 1, 1};
30     unsigned long long prime[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
31     unsigned long long composite[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
32
33     // primeIterator(a, prime, composite, arrayLength);
34     isPrimeAssembly(a, prime, composite, arrayLength);
35
36     printf("Input Array elements were: ");
37     for(int i = 0; i < arrayLength; i++)
38     {
39         printf("%d ", a[i]);
40     }
41     printf("\n");
42
43     printf("Prime Array elements are: ");
44     for(int i = 0; i < arrayLength; i++)
45     {
46         printf("%d ", prime[i]);
47     }
48     printf("\n");
49
50     printf("Composite Array elements are: ");
51     for(int i = 0; i < arrayLength; i++)
52     {
53         printf("%d ", composite[i]);
54     }
55     printf("\n");
56 }

```
- Right Pane (Assembly Code):**

```

1 .globl isPrimeAssembly
2
3 isPrimeAssembly: // Same as primeIterator()
4 // Your code for iterating through arrays here
5 // The Base addresses of arrays in X0 - X2, length in X3
6 // X0 has the base address of Original array
7 // X1 has the base address of Prime array
8 // X2 has the base address of Composite array
9 // X3 has the length of the Original array
10 nop //first line of execution when called by main.c
11 mov x20, x30 // Save the return register for later
12 mov x4, #0 // Initialize as x4=index=0
13
14 iterate:
15 ldr x19, [x0, x4, lsl #3] // Load value from original array (x0)
16 // x19 is going to be our n value
17 bl isPrime // Call isPrime sub-function, branch to label
18
19 cbz x6, storeComposite // If the returnZero label is used and
20 str x19, [x1, x4, lsl #3] // Else store x19 into the Prime array
21 b nextIndex // Branch to calculate next index
22
23 storeComposite:
24 str x19, [x2, x4, lsl #3] // Store x19 into the Composite array
25 b nextIndex // Branch to calculate next index
26
27 nextIndex:
28 add x4, x4, #1 // Index=x4+1, increase index by 1
29 cmp x3, x4 // Compare length of original array to altered x4
30 b.ne iterate // Continue iterating until all array values are
31 mov x30, x20 // Reload value of branch to x30
32 ret // Return to C code

```
- Bottom Pane (Terminal Output):**

```

• [audnle@audnle ECE 331 Project 1]$ aarch64-linux-gnu-gcc main.c isPrimeAssembly.s -g -o isPrime -no-pie -O0
• [audnle@audnle ECE 331 Project 1]$ qemu-aarch64 -L /usr/aarch64-linux-gnu/ ./isPrime
Input Array elements were: 7 16 23 40 11 39 37 10 2 18 44 83 87 5 6 11
Prime Array elements are: 7 0 23 0 11 0 37 0 2 0 0 83 0 5 0 11
Composite Array elements are: 0 16 0 40 0 39 0 10 0 18 44 0 87 0 6 0
• [audnle@audnle ECE 331 Project 1]$

```