Exploring the assembly for fun and profit

—

By Shivanjan Chakravorty

Assembly the language of Gods

What is Assembly?

In computer programming, **assembly language** (or **assembler language**), often abbreviated **asm**, is any low-level programming language in which there is a very strong correspondence between the instructions in the language and the architecture's machine code instructions.

Who needs it?

To be frank
nobody does!

Okay so why am I even attending this?

Assembly, a want not a need!

Apparently we got far enough with high level programming languages that it takes just one line to write a “Hello, world!” program.

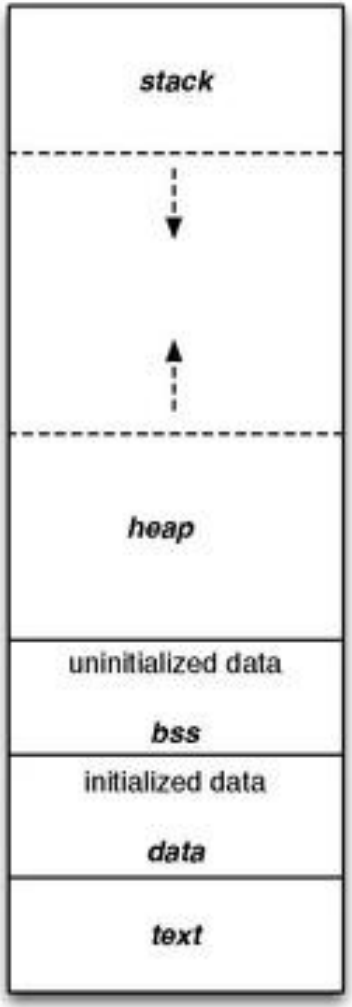
But as I said it’s more a matter of want than need!

Most of the hackers and reverse engineering experts keep assembly in their skillset.

It is heavily used by people dealing in embedded programming.

It’s a cool thing if you want to show off.

Revisiting our boring syllabus because it is necessary



- Stack takes care of the function calls, recursion, growing memory operations. Unlike stack in our day to day programming the stack in your memory rather grows downwards.
- Heap is the region where dynamic memory allocation happens. So if you have ever used malloc, calloc or free you have changed bytes of memory in the heap.
- BSS aka Block Started By Symbol this is where all your uninitialized variables go. E.g. int i, static int j
- Data segment holds all the initialized variables. E.g. int i =2, global int j = 3
- Text segment is a read only part that holds certain executable instructions. It also acts as border to prevent stack overflow

64-bit register	Lower 32 bits	Lower 16 bits	Lower 8 bits
rax	eax	ax	al
rbx	ebx	bx	bl
rcx	ecx	cx	cl
rdx	edx	dx	dl
rsi	esi	si	sil
rdi	edi	di	dil
rbp	ebp	bp	bpl
rsp	esp	sp	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10b
r11	r11d	r11w	r11b
r12	r12d	r12w	r12b
r13	r13d	r13w	r13b
r14	r14d	r14w	r14b
r15	r15d	r15w	r15b

Every processor has a different architecture and word length and here the registers you see

Dealing with any processor requires a basic idea of where to store and read things.

Computers may seem all fast and evolved today but deep down it is always bunch of register read writes that make the giant programs move.

The Experiment

Materials

Things you could just apt install

- NASM assembler
- GCC
- Radare2 (Optional)

Thank you.