Why Assembly Language?

How computer run your application?

```c
#include <stdio.h>

int main(){
printf("Hello World");
}
```

https://godbolt.org

```
.LC0:
        .ascii  "Hello World\000"
main:
        stmfd   sp!, {fp, lr}
        add     fp, sp, #4
        ldr     r0, .L2
        bl      printf
        mov     r3, #0
        mov     r0, r3
        ldmfd   sp!, {fp, pc}
.L2:
        .word   .LC0
```

C++ source #1

```
1  int main(){
2  }
3
```

C++ source #1

```
1  int main(){
2  int i;
3  }
4
```

C++ source #1

```
1  int main(){
2  int i;
3  i=i+1;
4  }
5
```

Local Variable Types

```c
int checksum_v1(int *data)
{
  char i;
  int sum=0;

  for (i=0; i<64; i++)
  {
    sum += data[i];
  }
  return sum;
}
```

```
checksum_v1
        MOV     r2,r0                   ; r2 = data
        MOV     r0,#0                   ; sum = 0
        MOV     r1,#0                   ; i = 0
checksum_v1_loop
        LDR     r3,[r2,r1,LSL #2]       ; r3 = data[i]
        ADD     r1,r1,#1                ; r1 = i+1
        AND     r1,r1,#0xff             ; i = (char)r1
        CMP     r1,#0x40                ; compare i, 64
        ADD     r0,r3,r0                ; sum += r3
        BCC     checksum_v1_loop        ; if (i<64) loop
        MOV     pc,r14                  ; return sum
```

```c
int checksum_v5(int *data)
{
  unsigned int i;
  int sum=0;

  for (i=0; i<64; i++)
  {
    sum += *(data++);
  }
  return sum;
}
```

```
checksum_v5
        MOV     r2,r0               ; r2 = data
        MOV     r0,#0               ; sum = 0
        MOV     r1,#0               ; i = 0
checksum_v5_loop
        LDR     r3,[r2],#4          ; r3 = *(data++)
        ADD     r1,r1,#1            ; i++
        CMP     r1,#0x40            ; compare i, 64
        ADD     r0,r3,r0            ; sum += r3
        BCC     checksum_v5_loop    ; if (i<64) goto loop
        MOV     pc,r14              ; return sum
```

Loops with a Fixed Number of Iterations

```c
int checksum_v6(int *data)
{
  unsigned int i;
  int sum=0;

  for (i=64; i!=0; i--)
  {
    sum += *(data++);
  }
  return sum;
}
```

```
checksum_v6
        MOV     r2,r0               ; r2 = data
        MOV     r0,#0               ; sum = 0
        MOV     r1,#0x40            ; i = 64
checksum_v6_loop
        LDR     r3,[r2],#4          ; r3 = *(data++)
        SUBS    r1,r1,#1            ; i-- and set flags
        ADD     r0,r3,r0            ; sum += r3
        BNE     checksum_v6_loop    ; if (i!=0) goto loop
        MOV     pc,r14              ; return sum
```

Loop Unrolling

```
int checksum_v9(int *data, unsigned int N)
{
  int sum=0;
  do
  {
    sum += *(data++);
    sum += *(data++);
    sum += *(data++);
    sum += *(data++);
    N -= 4;
  } while ( N!=0);
  return sum;
}
```

```
checksum_v9
        MOV r2,#0          ; sum = 0
checksum_v9_loop
        LDR     r3,[r0],#4         ; r3 = *(data++)
        SUBS    r1,r1,#4           ; N -= 4 & set flags
        ADD     r2,r3,r2           ; sum += r3
        LDR     r3,[r0],#4         ; r3 = *(data++)
        ADD     r2,r3,r2           ; sum += r3
        LDR     r3,[r0],#4         ; r3 = *(data++)
        ADD     r2,r3,r2           ; sum += r3
        LDR     r3,[r0],#4         ; r3 = *(data++)
        ADD     r2,r3,r2           ; sum += r3
        BNE     checksum_v9_loop   ; if (N!=0) goto loop
        MOV     r0,r2              ; r0 = sum
        MOV     pc,r14             ; return r0
```

Summery

- You can improve program performance and figure out the problems if you know the algorithm and data structure

- You can improve even more and identify the problems if you know the assembly language