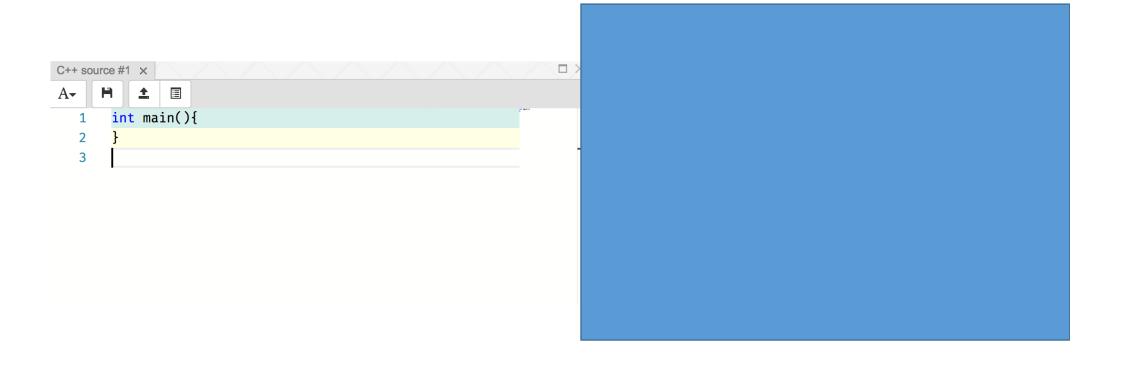
Why Assembly Language?

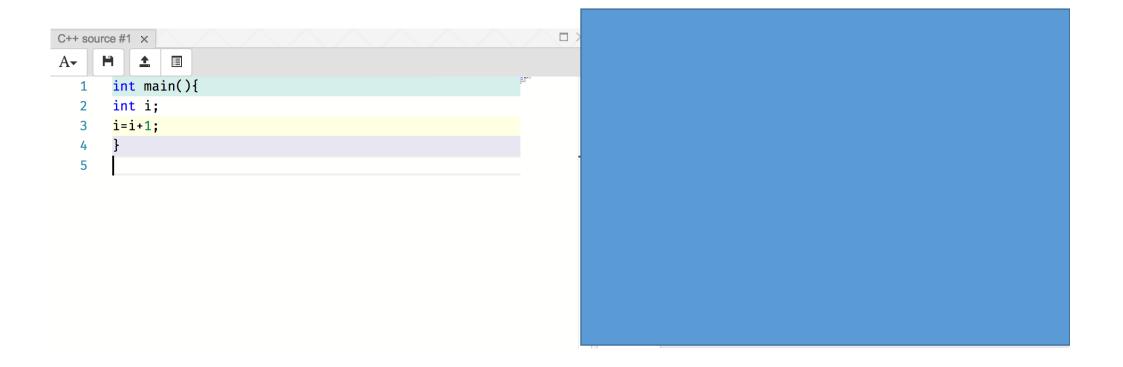
How computer run your application?

```
1 #include <stdio.h>
2
3 int main(){
4 printf("Hello World");
5 }
```

```
1 .LC0:
          .ascii "Hello World\000"
3 main:
         stmfd sp!, {fp, lr}
4
         add fp, sp, #4
5
         ldr r0, .L2
6
                 printf
         bl
         mov r3, #0
8
         mov r0, r3
9
                sp!, {fp, pc}
         ldmfd
10
11 .L2:
12
                 .LC0
          .word
```







Local Variable Types

```
checksum v1
int checksum v1(int *data)
                                      MOV
                                                                  ; r2 = data
                                              r2,r0
                                      MOV
                                              r0,#0
                                                                  sum = 0
 char i:
                                      MOV
                                              r1,#0
                                                                  : i = 0
 int sum = 0;
                              checksum v1 loop
                                      LDR
                                              r3,[r2,r1,LSL #2] ; r3 = data[i]
 for (i = 0; i < 64; i++)
                                      ADD
                                              r1,r1,#1
                                                            ; r1 = i+1
                                                                  ; i = (char)r1
                                      AND
                                              rl,rl,#0xff
   sum += data[i];
                                      CMP
                                              r1,#0x40
                                                                  ; compare i, 64
                                              r0,r3,r0
                                      ADD
                                                                  ; sum += r3
 return sum;
                                      BCC
                                              checksum v1 loop
                                                                  ; if (i<64) loop
                                      MOV
                                              pc,r14
                                                                  ; return sum
```

```
checksum v5
                                   ; r2 = data
       MOV
                r2,r0
        MOV
                r0,#0
                                   ; sum = 0
        MOV
                r1,#0
                                   i = 0
checksum v5 loop
                                   ; r3 = *(data++)
       LDR
                r3,[r2],#4
        ADD
                r1,r1,#1
                                   ; j++
        CMP
                r1,#0x40
                                   ; compare i, 64
       ADD
                r0,r3,r0
                                   : sum += r3
                checksum v5 loop ; if (i<64) goto loop
        BCC
       MOV
                pc,r14
                                   ; return sum
```

Loops with a Fixed Number of Iterations

```
int checksum v6(int *data)
 unsigned int i;
 int sum=0;
 for (i=64; i!=0; i--)
   sum += *(data++);
                       checksum v6
                               MOV
                                        r2,r0
                                                          ; r2 = data
 return sum;
                               MOV
                                        r0,#0
                                                           sum = 0
                                        r1,#0x40
                               MOV
                                                           i = 64
                       checksum v6 loop
                                                          ; r3 = *(data++)
                               LDR
                                        r3,[r2],#4
                               SUBS
                                        r1,r1,#1
                                                           ; i-- and set flags
                               ADD
                                        r0,r3,r0
                                                          : sum += r3
                                        checksum v6 loop ; if (i!=0) goto loop
                               BNE
                               MOV
                                        pc,r14
                                                           : return sum
```

Loop Unrolling

```
int checksum v9(int *data, unsigned int N)
 int sum=0;
                   checksum v9
                           MOV r2.\#0 : sum = 0
                    checksum v9 loop
 do
                                        r3,[r0],#4; r3 = *(data++)
                           LDR
   sum += *(data++);
                           SUBS
                                        r1.r1.#4
                                                          : N -= 4 & set flags
   sum += *(data++);
                           ADD
                                        r2.r3.r2
                                                          : sum += r3
   sum += *(data++);
                                                          : r3 = *(data++)
                           LDR
                                        r3,[r0],#4
   sum += *(data++);
                           ADD
                                        r2.r3.r2
                                                          : sum += r3
   N = 4:
                                                          : r3 = *(data++)
                           LDR
                                        r3,[r0],#4
 } while ( N!=0);
                           ADD
                                        r2.r3.r2
                                                          : sum += r3
                                                          ; r3 = *(data++)
 return sum;
                           LDR
                                        r3,[r0],#4
                           ADD
                                        r2,r3,r2
                                                          : sum += r3
                                        checksum v9 loop
                                                          ; if (N!=0) goto loop
                           BNE
                           MOV
                                        r0.r2
                                                          : r0 = sum
                                                          ; return r0
                           MOV
                                        pc.r14
```

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