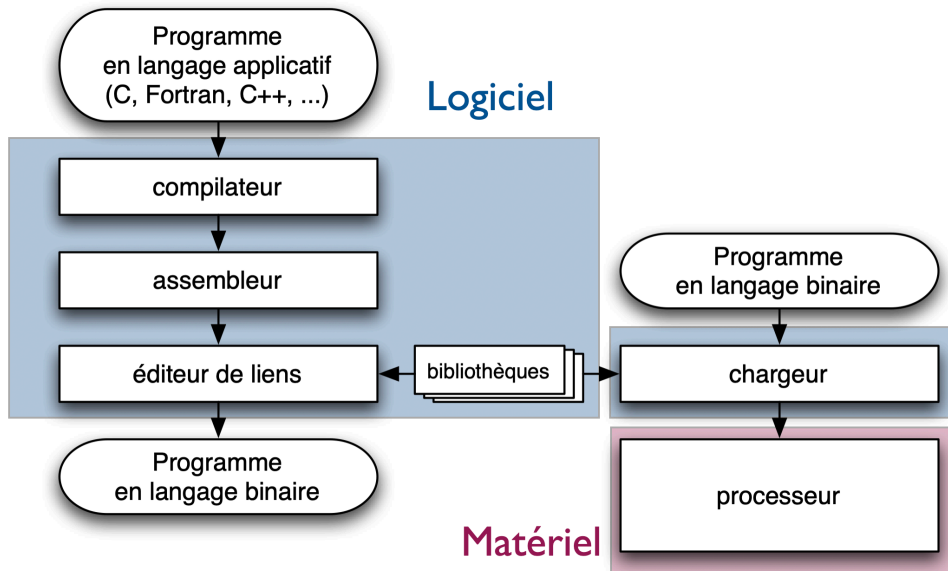


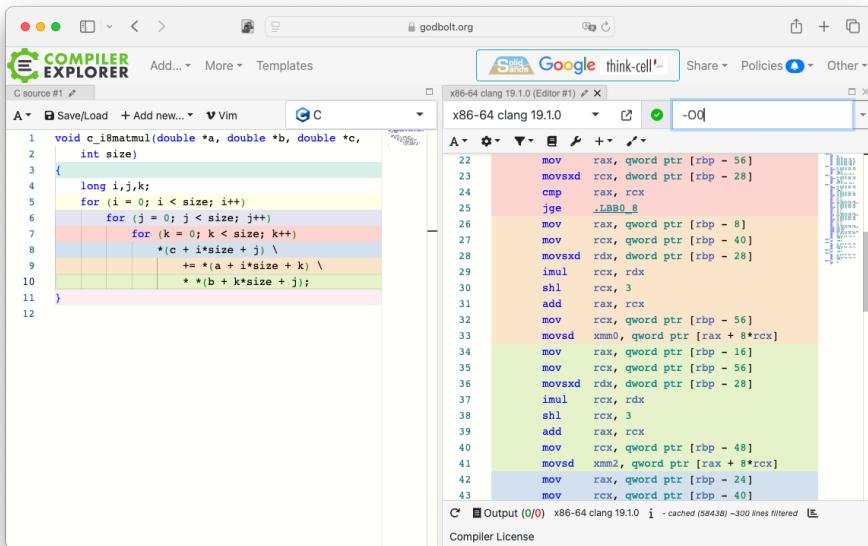
Introduction au développement logiciel

Langage machine et langage d'assemblage

Georges-André Silber, Centre de recherche en informatique

Mines Paris — PSL, octobre 2025





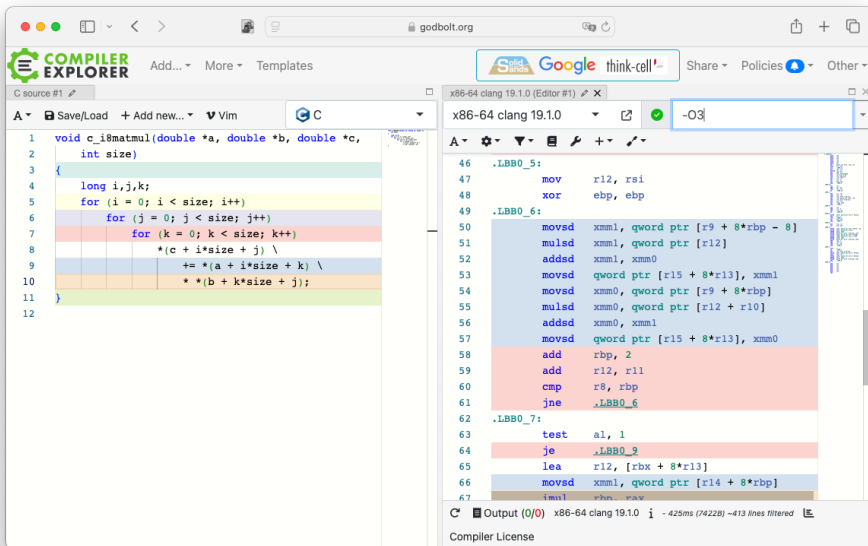
The screenshot shows the Compiler Explorer interface. The left pane displays the C source code for a function `c_i8matmul` that performs a naive matrix multiplication of two 8x8 integer matrices. The right pane shows the assembly output for x86-64 clang 19.1.0 with the `-O0` optimization level. The assembly is unoptimized, showing a direct translation of the C code's nested loops and arithmetic operations into x86-64 instructions.

```
1 void c_i8matmul(double *a, double *b, double *c,  
2 int size)  
3 {  
4     long i,j,k;  
5     for (i = 0; i < size; i++)  
6         for (j = 0; j < size; j++)  
7             for (k = 0; k < size; k++)  
8                 *(c + i*size + j) \  
9                 += *(a + i*size + k) \  
10                 * *(b + k*size + j);  
11 }  
12
```

```
22 mov     rax, qword ptr [rbp - 56]  
23 movsxd rcx, dword ptr [rbp - 28]  
24 cmp     rax, rcx  
25 jge     .LBB0_8  
26 mov     rax, qword ptr [rbp - 8]  
27 mov     rcx, qword ptr [rbp - 40]  
28 movsxd rdx, dword ptr [rbp - 28]  
29 imul    rcx, rdx  
30 shl     rcx, 3  
31 add     rax, rcx  
32 mov     rcx, qword ptr [rbp - 56]  
33 movsd   xmm0, qword ptr [rax + 8*rcx]  
34 mov     rax, qword ptr [rbp - 16]  
35 mov     rcx, qword ptr [rbp - 56]  
36 movsxd rdx, dword ptr [rbp - 28]  
37 imul    rcx, rdx  
38 shl     rcx, 3  
39 add     rax, rcx  
40 mov     rcx, qword ptr [rbp - 48]  
41 movsd   xmm2, qword ptr [rax + 8*rcx]  
42 mov     rax, qword ptr [rbp - 24]  
43 mov     rcx, qword ptr [rbp - 40]
```

Output (0/0) x86-64 clang 19.1.0 i - cached (5843B) ~300 lines filtered

Compiler License



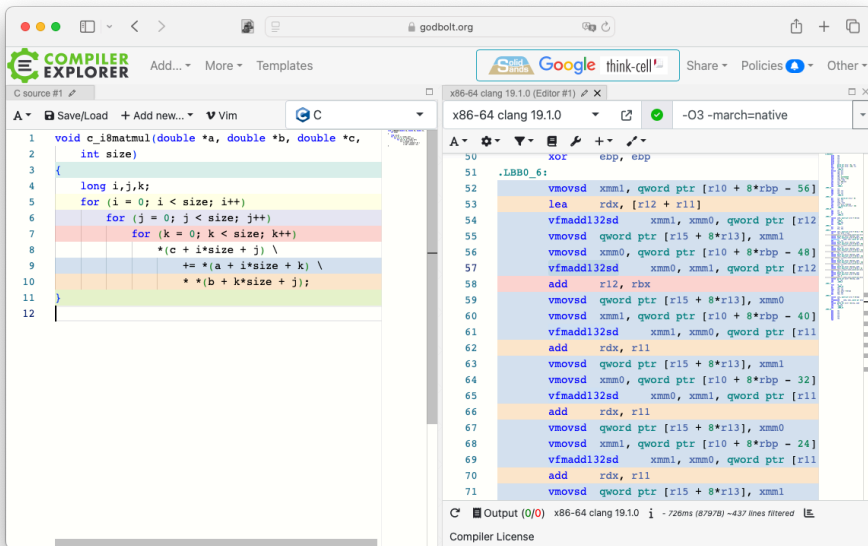
The screenshot shows the Compiler Explorer interface. The left pane displays C source code for a matrix multiplication function. The right pane shows the generated assembly code for x86-64 using clang 19.1.0 with the -O3 optimization level. The assembly includes labels like .LBB0_5 and .LBB0_6, and instructions such as mov, xor, movsd, mulsd, addsd, and cmp.

```
1 void c_i8matmul(double *a, double *b, double *c,  
2 int size)  
3 {  
4     long i,j,k;  
5     for (i = 0; i < size; i++)  
6         for (j = 0; j < size; j++)  
7             for (k = 0; k < size; k++)  
8                 *(c + i*size + j) \  
9                 += *(a + i*size + k) \  
10                * *(b + k*size + j);  
11 }  
12
```

```
46 .LBB0_5:  
47     mov     r12, rsi  
48     xor     ebp, ebp  
49 .LBB0_6:  
50     movsd   xmm1, qword ptr [r9 + 8*rbp - 8]  
51     mulsd   xmm1, qword ptr [r12]  
52     addsd   xmm1, xmm0  
53     movsd   qword ptr [r15 + 8*r13], xmm1  
54     movsd   xmm0, qword ptr [r9 + 8*rbp]  
55     mulsd   xmm0, qword ptr [r12 + r10]  
56     addsd   xmm0, xmm1  
57     movsd   qword ptr [r15 + 8*r13], xmm0  
58     add     rbp, 2  
59     add     r12, r11  
60     cmp     r8, rbp  
61     jne     .LBB0_6  
62 .LBB0_7:  
63     test    al, 1  
64     je      .LBB0_9  
65     lea     r12, [rbx + 8*r13]  
66     movsd   xmm1, qword ptr [r14 + 8*rbp]  
67     imul    rbp, rax
```

Output (0/0) x86-64 clang 19.1.0 i - 425ms (74228) ~413 lines filtered

Compiler License



The screenshot displays the Godbolt compiler explorer interface. On the left, the C source code for a function `c_i8matmul` is shown, which performs a matrix multiplication. The code uses nested loops for `i`, `j`, and `k`, and includes pointer arithmetic to access elements in a 2D array. The right pane shows the generated assembly code for x86-64 using clang 19.1.0 with the `-O3 -march=native` optimization flags. The assembly includes instructions for stack frame setup, memory access, and vectorized operations using SSE registers (`xmm0`, `xmm1`). The bottom status bar indicates the compilation took 726ms and generated 437 lines of assembly.

```
1 void c_i8matmul(double *a, double *b, double *c,  
2 int size)  
3 {  
4     long i,j,k;  
5     for (i = 0; i < size; i++)  
6         for (j = 0; j < size; j++)  
7             for (k = 0; k < size; k++)  
8                 *(c + i*size + j) \  
9                 += *(a + i*size + k) \  
10                * *(b + k*size + j);  
11 }  
12
```

```
50     xor     ebp, ebp  
51     .LBB_6:  
52     vmovsd  xmm1, qword ptr [r10 + 8*rbp - 56]  
53     lea     rdx, [r12 + r11]  
54     vfmadd132sd  xmm1, xmm0, qword ptr [r12  
55     vmovsd  qword ptr [r15 + 8*r13], xmm1  
56     vmovsd  xmm0, qword ptr [r10 + 8*rbp - 48]  
57     vfmadd132sd  xmm0, xmm1, qword ptr [r12  
58     add     r12, rbx  
59     vmovsd  qword ptr [r15 + 8*r13], xmm0  
60     vmovsd  xmm1, qword ptr [r10 + 8*rbp - 40]  
61     vfmadd132sd  xmm1, xmm0, qword ptr [r11  
62     add     rdx, r11  
63     vmovsd  qword ptr [r15 + 8*r13], xmm1  
64     vmovsd  xmm0, qword ptr [r10 + 8*rbp - 32]  
65     vfmadd132sd  xmm0, xmm1, qword ptr [r11  
66     add     rdx, r11  
67     vmovsd  qword ptr [r15 + 8*r13], xmm0  
68     vmovsd  xmm1, qword ptr [r10 + 8*rbp - 24]  
69     vfmadd132sd  xmm1, xmm0, qword ptr [r11  
70     add     rdx, r11  
71     vmovsd  qword ptr [r15 + 8*r13], xmm1
```

Output (0/0) x86-64 clang 19.1.0 i - 726ms (87978) -437 lines filtered

Compiler License