Tugas 1 Praktikum Sistem Komputer dan Jaringan

Nama: Muhammad Argya Vityasy
NIM: 23/522547/PA/22475
Kelas: Praktikum SKJ KOM A

1.6.1 GitHub

Muhammad Argya Vityasy's GitHub for Praktikum SKJ

1.6.2 C++ to Assembly

1. Simple C++ program to add 2 numbers

The code is available in the GitHub page at Assignment1/CtoAssembly/add.cpp

2. Compile the code

It makes the 'add' binary file, an executable file created after we compile our code

3. Disassembled code documentation

Assignment1 > CtoAssembly > ASM add.asm

```
1
2   add:    file format mach-o 64-bit x86-64
3
4   Disassembly of section __TEXT,__text:
5
6   0000000100000f60 <_main>:
7   100000f60: 55                          pushq   %rbp
8   100000f61: 48 89 e5                    movq    %rsp, %rbp
9   100000f64: 48 83 ec 10                 subq    $16, %rsp
10  100000f68: c7 45 fc 00 00 00 00        movl    $0, -4(%rbp)
11  100000f6f: c7 45 f8 05 00 00 00        movl    $5, -8(%rbp)
12  100000f76: c7 45 f4 17 00 00 00        movl    $23, -12(%rbp)
13  100000f7d: 8b 45 f8                    movl    -8(%rbp), %eax
14  100000f80: 03 45 f4                    addl    -12(%rbp), %eax
15  100000f83: 89 45 f0                    movl    %eax, -16(%rbp)
16  100000f86: 8b 75 f0                    movl    -16(%rbp), %esi
17  100000f89: 48 8b 3d 78 00 00 00        movq    120(%rip), %rdi        ## 0x100001008 <__ZNSt3__14coutE+0x100001008>
18  100000f90: e8 08 00 00 00              callq   0x100000f9d <__ZNSt3__14coutE+0x100000f9d>
19  100000f95: 31 c0                       xorl    %eax, %eax
20  100000f97: 48 83 c4 10                 addq    $16, %rsp
21  100000f9b: 5d                          popq    %rbp
22  100000f9c: c3                          retq
23
24  Disassembly of section __TEXT,__stubs:
25
26  0000000100000f9d <__stubs>:
27  100000f9d: ff 25 5d 00 00 00           jmpq    *93(%rip)              ## 0x100001000 <__ZNSt3__14coutE+0x100001000>
28
```

Figure 1: add.asm, the assembled C++ code

4. Makefile

The Makefile code is available in the GitHub page at Assignment1/CtoAssembly/Makefile

5. Output of the ’make’ command

```
miapalovaara@Mias-MacBook-Air-2 AssemblytoC % cd ../C*
miapalovaara@Mias-MacBook-Air-2 CtoAssembly % make clean
rm -f add add.asm
miapalovaara@Mias-MacBook-Air-2 CtoAssembly % make
g++  -o add add.cpp
miapalovaara@Mias-MacBook-Air-2 CtoAssembly % make run
./add
28%
miapalovaara@Mias-MacBook-Air-2 CtoAssembly % make dump
objdump -d add > add.asm
miapalovaara@Mias-MacBook-Air-2 CtoAssembly %
```

Figure 2: Output of the ’make’ command

1.6.3 Assembly to C++

1. Analyzing the provided code

add.cpp Assignment1 | multiply.asm | multiply.cpp | Makefile | multiply1.asm | add.asm | add.cpp .../CtoAssembly

Assignment1 > AssemblytoC > multiply.asm

```
1   section .data       ; our variable for the program
2       num1 dq 5       ; create a variable called num1, define quadword, 8 bytes on a x86 64-bit system. 64-bit integer with initial value of 5
3       num2 dq 10      ; same thing
4       result dq 0
5
6   section .text       ; the code segment
7       global _main    ; default entry point for nasm macho64
8       default rel     ; the default addressing in 64-bit mode
9
10  _main:
11      mov rax, [num1]       ; load the value from the num1 memory to register
12      imul rax, [num2]      ; multiply the value from the num2 memory with register
13      mov [result], rax     ; store the value from the register to the memory of result
14
15      ; Exit the program
16      mov rax, 0x2000001    ; syscall number for exit in MacOS
17      xor rdi, rdi          ; xor ing by itself results to 0, status code 0 (like return 0 in C++)
18      syscall               ; initiate the call to the OS
19
```

Figure 3: Code documentation with explanation per line

The code doesn’t really output anything yet, i’ve tried to compile and link it with these 2 commands:

`nasm -f macho64 multiply.asm -o multiply.o`

It creates the ’multiply.o’ file after the compile process

After the compile process, we link it with this link command:

```
ld -macos_version_min 10.7.0 -o multiply multiply.o -lSystem -L
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/lib
```

It creates the executable ’multiply’ file, and the program outputs nothing when i run ’./multiply’.

The code is also available in the GitHub page at Assignment1/AssemblytoC/multiply.asm

2. Writing the equivalent C++ code

The code is available in the GitHub page at Assignment1/AssemblytoC/multiply.cpp

3. Writing the Makefile

The code is available in the GitHub page at Assignment1/AssemblytoC/Makefile

```
miapalovaara@Mias-MacBook-Air-2 AssemblytoC % make clean
rm -f multiply multiply1.asm
miapalovaara@Mias-MacBook-Air-2 AssemblytoC % make
g++  -o multiply multiply.cpp
miapalovaara@Mias-MacBook-Air-2 AssemblytoC % make run
./multiply
50%
miapalovaara@Mias-MacBook-Air-2 AssemblytoC % make dump
objdump -d multiply > multiply1.asm
miapalovaara@Mias-MacBook-Air-2 AssemblytoC %
```

Figure 4: The output of the 'make' command

Assignment1 > AssemblytoC > multiply1.asm

```
1
2   multiply:   file format mach-o 64-bit x86-64
3
4   Disassembly of section __TEXT,__text:
5
6   0000000100000f60 <_main>:
7   100000f60: 55                          pushq   %rbp
8   100000f61: 48 89 e5                    movq    %rsp, %rbp
9   100000f64: 48 83 ec 10                 subq    $16, %rsp
10  100000f68: c7 45 fc 00 00 00 00        movl    $0, -4(%rbp)
11  100000f6f: c7 45 f8 05 00 00 00        movl    $5, -8(%rbp)
12  100000f76: c7 45 f4 0a 00 00 00        movl    $10, -12(%rbp)
13  100000f7d: 8b 45 f8                    movl    -8(%rbp), %eax
14  100000f80: 0f af 45 f4                 imull   -12(%rbp), %eax
15  100000f84: 89 45 f0                    movl    %eax, -16(%rbp)
16  100000f87: 8b 75 f0                    movl    -16(%rbp), %esi
17  100000f8a: 48 8b 3d 77 00 00 00        movq    119(%rip), %rdi        ## 0x100001008 <__ZNSt3__14coutE+0x100001008>
18  100000f91: e8 08 00 00 00              callq   0x100000f9e <__ZNSt3__14coutE+0x100000f9e>
19  100000f96: 31 c0                       xorl    %eax, %eax
20  100000f98: 48 83 c4 10                 addq    $16, %rsp
21  100000f9c: 5d                          popq    %rbp
22  100000f9d: c3                          retq
23
24  Disassembly of section __TEXT,__stubs:
25
26  0000000100000f9e <__stubs>:
27  100000f9e: ff 25 5c 00 00 00           jmpq    *92(%rip)              ## 0x100001000 <__ZNSt3__14coutE+0x100001000>
28
```

Figure 5: The multiply1.asm, assembled C++ file using the 'make dump' command

End of Document, Thank You