Lab 1

Riya Patel

CSC 345-01: Operating Systems

2/2/2022

C code (factorial.c):

```
#include <stdio.h>
int main(){
int number=4;
printf("Factorial of %d is: %d", number, factorial(number));
}

int factorial(int number){
int i, result = 1;
if (number==0)
     return 0;
else {
     for(i=1;i<=number;i++){
          result=result*i;
     }
}
return result;
}
```

Compiler code (factorial.s):

```
     .file "factorial.c"
     .text
     .section   .rodata
.LC0:
     .string    "Factorial of %d is: %d"
     .text
     .globl     main
     .type main, @function
main:
.LFB0:
     .cfi_startproc
     endbr64
```

```
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$16, %rsp
	movl	$4, -4(%rbp)
	movl	-4(%rbp), %eax
	movl	%eax, %edi
	movl	$0, %eax
	call	factorial
	movl	%eax, %edx
	movl	-4(%rbp), %eax
	movl	%eax, %esi
	leaq	.LC0(%rip), %rdi
	movl	$0, %eax
	call	printf@PLT
	movl	$0, %eax
	leave
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
	.size	main, .-main
	.globl	factorial
	.type	factorial, @function
factorial:
.LFB1:
	.cfi_startproc
	endbr64
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	movl	%edi, -20(%rbp)
	movl	$1, -4(%rbp)
	cmpl	$0, -20(%rbp)
	jne	.L4
	movl	$0, %eax
	jmp	.L5
.L4:
	movl	$1, -8(%rbp)
	jmp	.L6
.L7:
	movl	-4(%rbp), %eax
	imull	-8(%rbp), %eax
	movl	%eax, -4(%rbp)
	addl	$1, -8(%rbp)
.L6:
	movl	-8(%rbp), %eax
	cmpl	-20(%rbp), %eax
	jle	.L7
	movl	-4(%rbp), %eax
.L5:
	popq	%rbp
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE1:
	.size	factorial, .-factorial
	.ident	"GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
	.section	.note.GNU-stack,"",@progbits
	.section	.note.gnu.property,"a"
```

```
        .align 8
        .long  1f - 0f
        .long  4f - 1f
        .long  5
0:
        .string     "GNU"
1:
        .align 8
        .long  0xc0000002
        .long  3f - 2f
2:
        .long  0x3
3:
        .align 8
4:
```

Disassembly code (factorial_dump.txt):

```
factorial:     file format elf64-x86-64


Disassembly of section .init:

0000000000001000 <_init>:
    1000:  f3 0f 1e fa           endbr64
    1004:  48 83 ec 08           sub    $0x8,%rsp
    1008:  48 8b 05 d9 2f 00 00  mov    0x2fd9(%rip),%rax        # 3fe8 <__gmon_start__>
    100f:  48 85 c0              test   %rax,%rax
    1012:  74 02                 je     1016 <_init+0x16>
    1014:  ff d0                 callq  *%rax
    1016:  48 83 c4 08           add    $0x8,%rsp
    101a:  c3                    retq
```

```
Disassembly of section .plt:

0000000000001020 <.plt>:
    1020:  ff 35 9a 2f 00 00     pushq  0x2f9a(%rip)        # 3fc0 <_GLOBAL_OFFSET_TABLE_+0x8>
    1026:  f2 ff 25 9b 2f 00 00 bnd jmpq *0x2f9b(%rip)        # 3fc8
<_GLOBAL_OFFSET_TABLE_+0x10>
    102d:  0f 1f 00              nopl   (%rax)
    1030:  f3 0f 1e fa           endbr64
    1034:  68 00 00 00 00        pushq  $0x0
    1039:  f2 e9 e1 ff ff ff     bnd jmpq 1020 <.plt>
    103f:  90                    nop

Disassembly of section .plt.got:

0000000000001040 <__cxa_finalize@plt>:
    1040:  f3 0f 1e fa           endbr64
    1044:  f2 ff 25 ad 2f 00 00 bnd jmpq *0x2fad(%rip)        # 3ff8
<__cxa_finalize@GLIBC_2.2.5>
    104b:  0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)

Disassembly of section .plt.sec:

0000000000001050 <printf@plt>:
    1050:  f3 0f 1e fa           endbr64
    1054:  f2 ff 25 75 2f 00 00 bnd jmpq *0x2f75(%rip)        # 3fd0 <printf@GLIBC_2.2.5>
    105b:  0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)

Disassembly of section .text:

0000000000001060 <_start>:
    1060:  f3 0f 1e fa           endbr64
    1064:  31 ed                 xor    %ebp,%ebp
    1066:  49 89 d1             mov    %rdx,%r9
    1069:  5e                   pop    %rsi
    106a:  48 89 e2             mov    %rsp,%rdx
    106d:  48 83 e4 f0          and    $0xfffffffffffffff0,%rsp
    1071:  50                   push   %rax
    1072:  54                   push   %rsp
    1073:  4c 8d 05 c6 01 00 00 lea    0x1c6(%rip),%r8        # 1240 <__libc_csu_fini>
    107a:  48 8d 0d 4f 01 00 00 lea    0x14f(%rip),%rcx        # 11d0 <__libc_csu_init>
    1081:  48 8d 3d c1 00 00 00 lea    0xc1(%rip),%rdi        # 1149 <main>
    1088:  ff 15 52 2f 00 00    callq  *0x2f52(%rip)        # 3fe0
<__libc_start_main@GLIBC_2.2.5>
    108e:  f4                   hlt
    108f:  90                   nop

0000000000001090 <deregister_tm_clones>:
    1090:  48 8d 3d 79 2f 00 00 lea    0x2f79(%rip),%rdi        # 4010 <__TMC_END__>
    1097:  48 8d 05 72 2f 00 00 lea    0x2f72(%rip),%rax        # 4010 <__TMC_END__>
    109e:  48 39 f8             cmp    %rdi,%rax
    10a1:  74 15                je     10b8 <deregister_tm_clones+0x28>
    10a3:  48 8b 05 2e 2f 00 00 mov    0x2f2e(%rip),%rax        # 3fd8
<_ITM_deregisterTMCloneTable>
    10aa:  48 85 c0             test   %rax,%rax
    10ad:  74 09                je     10b8 <deregister_tm_clones+0x28>
    10af:  ff e0                jmpq   *%rax
    10b1:  0f 1f 80 00 00 00 00 nopl   0x0(%rax)
    10b8:  c3                   retq
    10b9:  0f 1f 80 00 00 00 00 nopl   0x0(%rax)

00000000000010c0 <register_tm_clones>:
    10c0:  48 8d 3d 49 2f 00 00 lea    0x2f49(%rip),%rdi        # 4010 <__TMC_END__>
    10c7:  48 8d 35 42 2f 00 00 lea    0x2f42(%rip),%rsi        # 4010 <__TMC_END__>
    10ce:  48 29 fe             sub    %rdi,%rsi
    10d1:  48 89 f0             mov    %rsi,%rax
    10d4:  48 c1 ee 3f          shr    $0x3f,%rsi
    10d8:  48 c1 f8 03          sar    $0x3,%rax
    10dc:  48 01 c6             add    %rax,%rsi
    10df:  48 d1 fe             sar    %rsi
    10e2:  74 14                je     10f8 <register_tm_clones+0x38>
    10e4:  48 8b 05 05 2f 00 00 mov    0x2f05(%rip),%rax        # 3ff0
<_ITM_registerTMCloneTable>
    10eb:  48 85 c0             test   %rax,%rax
    10ee:  74 08                je     10f8 <register_tm_clones+0x38>
    10f0:  ff e0                jmpq   *%rax
    10f2:  66 0f 1f 44 00 00    nopw   0x0(%rax,%rax,1)
    10f8:  c3                   retq
    10f9:  0f 1f 80 00 00 00 00 nopl   0x0(%rax)

0000000000001100 <__do_global_dtors_aux>:
    1100:  f3 0f 1e fa          endbr64
    1104:  80 3d 05 2f 00 00 00 cmpb   $0x0,0x2f05(%rip)        # 4010 <__TMC_END__>
    110b:  75 2b                jne    1138 <__do_global_dtors_aux+0x38>
    110d:  55                   push   %rbp
    110e:  48 83 3d e2 2e 00 00 cmpq   $0x0,0x2ee2(%rip)        # 3ff8
<__cxa_finalize@GLIBC_2.2.5>
    1115:  00
    1116:  48 89 e5             mov    %rsp,%rbp
    1119:  74 0c                je     1127 <__do_global_dtors_aux+0x27>
    111b:  48 8b 3d e6 2e 00 00 mov    0x2ee6(%rip),%rdi        # 4008 <__dso_handle>
    1122:  e8 19 ff ff ff       callq  1040 <__cxa_finalize@plt>
    1127:  e8 64 ff ff ff       callq  1090 <deregister_tm_clones>
    112c:  c6 05 dd 2e 00 00 01 movb   $0x1,0x2edd(%rip)        # 4010 <__TMC_END__>
    1133:  5d                   pop    %rbp
    1134:  c3                   retq
    1135:  0f 1f 00             nopl   (%rax)
    1138:  c3                      retq
    1139:  0f 1f 80 00 00 00 00    nopl   0x0(%rax)

0000000000001140 <frame_dummy>:
    1140:  f3 0f 1e fa             endbr64
    1144:  e9 77 ff ff ff          jmpq   10c0 <register_tm_clones>

0000000000001149 <main>:
    1149:  f3 0f 1e fa             endbr64
    114d:  55                      push   %rbp                         # push the old base pointer
onto the stack
    114e:  48 89 e5                mov    %rsp,%rbp                    # copy the value of the stack
pointer to the base pointer
    1151:  48 83 ec 10             sub    $0x10,%rsp                   # allocate 16 bytes of space on
the stack
    1155:  c7 45 fc 04 00 00 00    movl   $0x4,-0x4(%rbp)              # %rbp-0x4 = 4 -- number = 4
    115c:  8b 45 fc                mov    -0x4(%rbp),%eax              # copy value of %rbp-0x4 to
%eax -- %eax = 4
    115f:  89 c7                   mov    %eax,%edi                    # copy %eax to %edi -- will
stay preserved due to register properties
    1161:  b8 00 00 00 00          mov    $0x0,%eax                    # copy 0 to %eax
    1166:  e8 1f 00 00 00          callq  118a <factorial>             # call factorial function
    116b:  89 c2                   mov    %eax,%edx                    # copy %eax to %edx
    116d:  8b 45 fc                mov    -0x4(%rbp),%eax              # %eax = -0x4(%rbp) -- %eax = 4
    1170:  89 c6                   mov    %eax,%esi                    # % copy %eax tp %esi -- %esi
=4
    1172:  48 8d 3d 8b 0e 00 00    lea    0xe8b(%rip),%rdi             # 2004 <_IO_stdin_used+0x4>   #
loads the address of the next instrcution - 0xe8b into %rdi -- address of the string
    1179:  b8 00 00 00 00          mov    $0x0,%eax                    # copy 0 to %eax
    117e:  e8 cd fe ff ff          callq  1050 <printf@plt>            # calls C library printf
function
    1183:  b8 00 00 00 00          mov    $0x0,%eax                    # copy 0 to %eax
    1188:  c9                      leaveq
    1189:  c3                      retq

000000000000118a <factorial>:
    118a:  f3 0f 1e fa             endbr64
    118e:  55                      push   %rbp                        # add stack pointer
    118f:  48 89 e5                mov    %rsp,%rbp                   # copy the value of the stack
pointer to the base pointer
    1192:  89 7d ec                mov    %edi,-0x14(%rbp)            # copy argument (number) to
%rbp-0x14
    1195:  c7 45 fc 01 00 00 00    movl   $0x1,-0x4(%rbp)             # copy 1 to %rbp-0x4 -- result
    119c:  83 7d ec 00             cmpl   $0x0,-0x14(%rbp)            # compare -- if number == 0
    11a0:  75 07                   jne    11a9 <factorial+0x1f>       # if number not equal jump
ahead to initialize i
    11a2:  b8 00 00 00 00          mov    $0x0,%eax                   # copy 0 to %eax
    11a7:  eb 22                   jmp    11cb <factorial+0x41>       # jump to return statement
    11a9:  c7 45 f8 01 00 00 00    movl   $0x1,-0x8(%rbp)             # copy 1 to %rbp-0x8 -- i = 1
    11b0:  eb 0e                   jmp    11c0 <factorial+0x36>       # jump to loop condition
validation (%eax = i)
    11b2:  8b 45 fc                mov    -0x4(%rbp),%eax             # copy %rbp-0x4 to %eax -- %eax
= result
    11b5:  0f af 45 f8             imul   -0x8(%rbp),%eax             # multiply %eax by i -- %eax =
result * i
    11b9:  89 45 fc                mov    %eax,-0x4(%rbp)             # copy %eax to %rbp-0x4 --
result = result * i
    11bc:  83 45 f8 01             addl   $0x1,-0x8(%rbp)             # add 1 to %rbp-0x8 -- i = i+1
(increment in for loop)
    11c0:  8b 45 f8                mov    -0x8(%rbp),%eax             # copy %rbp-0x8 to %eax -- %eax
= i
    11c3:  3b 45 ec                cmp    -0x14(%rbp),%eax            # i <= %rbp-0x14 -- 1<= number?
(for loop i<= number check)
```

```
    11c6:  7e ea                jle    11b2 <factorial+0x28>     # if %eax <= number than jump 
back to inside of loop
    11c8:  8b 45 fc             mov    -0x4(%rbp),%eax           # copy %rbp-0x4 to %eax -- %eax 
= result
    11cb:  5d                   pop    %rbp                      # pop stack pointer value
    11cc:  c3                   retq   
    11cd:  0f 1f 00             nopl   (%rax)

00000000000011d0 <__libc_csu_init>:
    11d0:  f3 0f 1e fa          endbr64 
    11d4:  41 57                push   %r15
    11d6:  4c 8d 3d db 2b 00 00 lea    0x2bdb(%rip),%r15         # 3db8 
<__frame_dummy_init_array_entry>
    11dd:  41 56                push   %r14
    11df:  49 89 d6             mov    %rdx,%r14
    11e2:  41 55                push   %r13
    11e4:  49 89 f5             mov    %rsi,%r13
    11e7:  41 54                push   %r12
    11e9:  41 89 fc             mov    %edi,%r12d
    11ec:  55                   push   %rbp
    11ed:  48 8d 2d cc 2b 00 00 lea    0x2bcc(%rip),%rbp         # 3dc0 
<__do_global_dtors_aux_fini_array_entry>
    11f4:  53                   push   %rbx
    11f5:  4c 29 fd             sub    %r15,%rbp
    11f8:  48 83 ec 08          sub    $0x8,%rsp
    11fc:  e8 ff fd ff ff       callq  1000 <_init>
    1201:  48 c1 fd 03          sar    $0x3,%rbp
    1205:  74 1f                je     1226 <__libc_csu_init+0x56>
    1207:  31 db                xor    %ebx,%ebx
    1209:  0f 1f 80 00 00 00 00 nopl   0x0(%rax)
    1210:  4c 89 f2             mov    %r14,%rdx
    1213:  4c 89 ee             mov    %r13,%rsi
    1216:  44 89 e7             mov    %r12d,%edi
    1219:  41 ff 14 df          callq  *(%r15,%rbx,8)
    121d:  48 83 c3 01          add    $0x1,%rbx
    1221:  48 39 dd             cmp    %rbx,%rbp
    1224:  75 ea                jne    1210 <__libc_csu_init+0x40>
    1226:  48 83 c4 08          add    $0x8,%rsp
    122a:  5b                   pop    %rbx
    122b:  5d                   pop    %rbp
    122c:  41 5c                pop    %r12
    122e:  41 5d                pop    %r13
    1230:  41 5e                pop    %r14
    1232:  41 5f                pop    %r15
    1234:  c3                   retq
    1235:  66 66 2e 0f 1f 84 00 data16 nopw %cs:0x0(%rax,%rax,1)
    123c:  00 00 00 00

0000000000001240 <__libc_csu_fini>:
    1240:  f3 0f 1e fa          endbr64
    1244:  c3                   retq

Disassembly of section .fini:

0000000000001248 <_fini>:
    1248:  f3 0f 1e fa          endbr64
    124c:  48 83 ec 08          sub    $0x8,%rsp
    1250:  48 83 c4 08          add    $0x8,%rsp
    1254:  c3                   retq
```