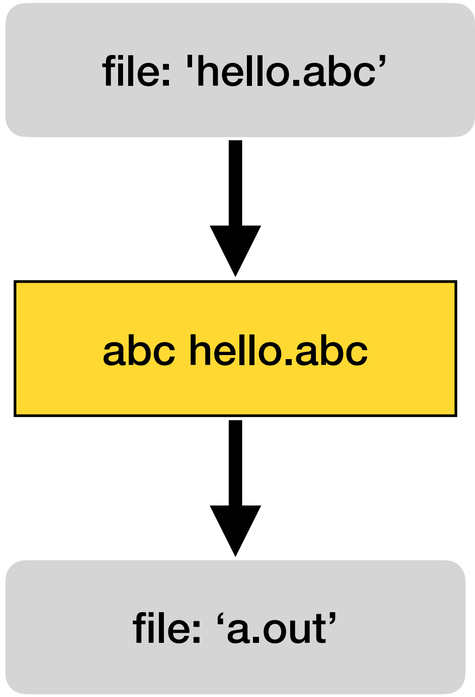
ABC by Example and Experiments (Top-Down Approach)

Look behind the sceens: Assembly Code, Object code, …
… and optimization flags

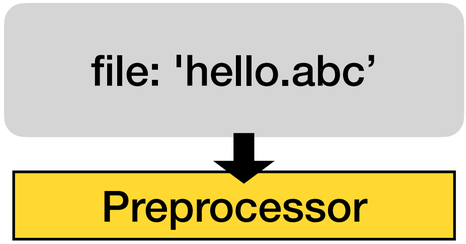
Phases of Translation

```
@ <stdio.hdr>

fn main()
{
    printf("hello, world!\n");
}
```



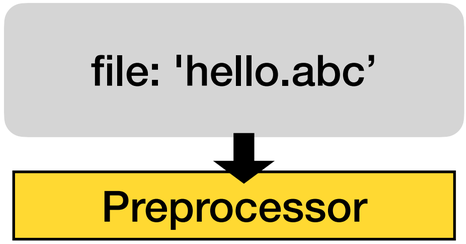
Phases of Translation

```
@ <stdio.hdr>

fn main()
{
    printf("hello, world!\n");
}
```

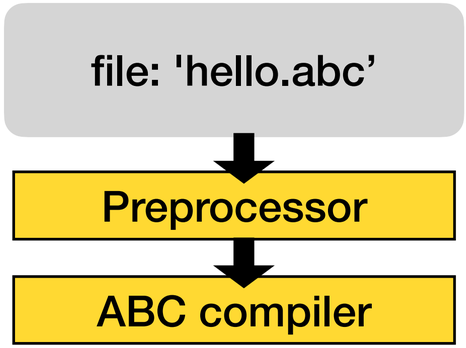


Phases of Translation

```
extern fn printf(fmt: -> char, ...): int;

fn main()
{
    printf("hello, world!\n");
}
```

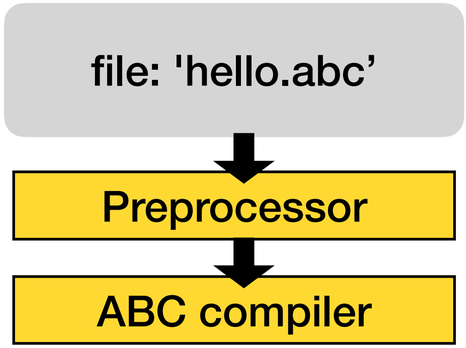


Phases of Translation



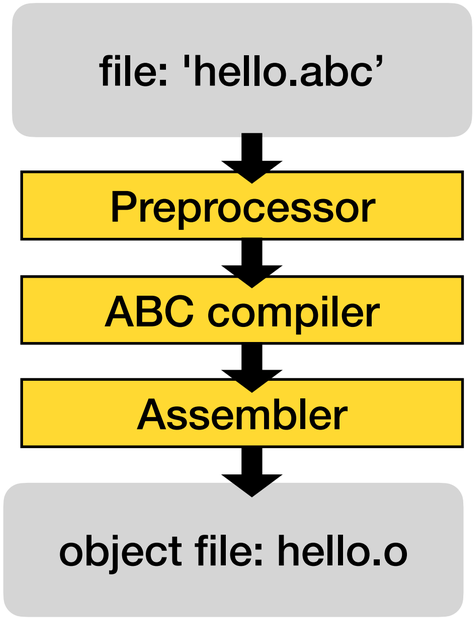
Phases of Translation

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .globl  _main
        .p2align        4, 0x90
_main:
        pushq   %rax
        leaq    _.L0(%rip), %rdi
        xorl    %eax, %eax
        callq   _printf
        popq    %rax
        retq

        .section        __TEXT,__const
_.L0:
        .asciz  "hello, world!\n"

.subsections_via_symbols
```

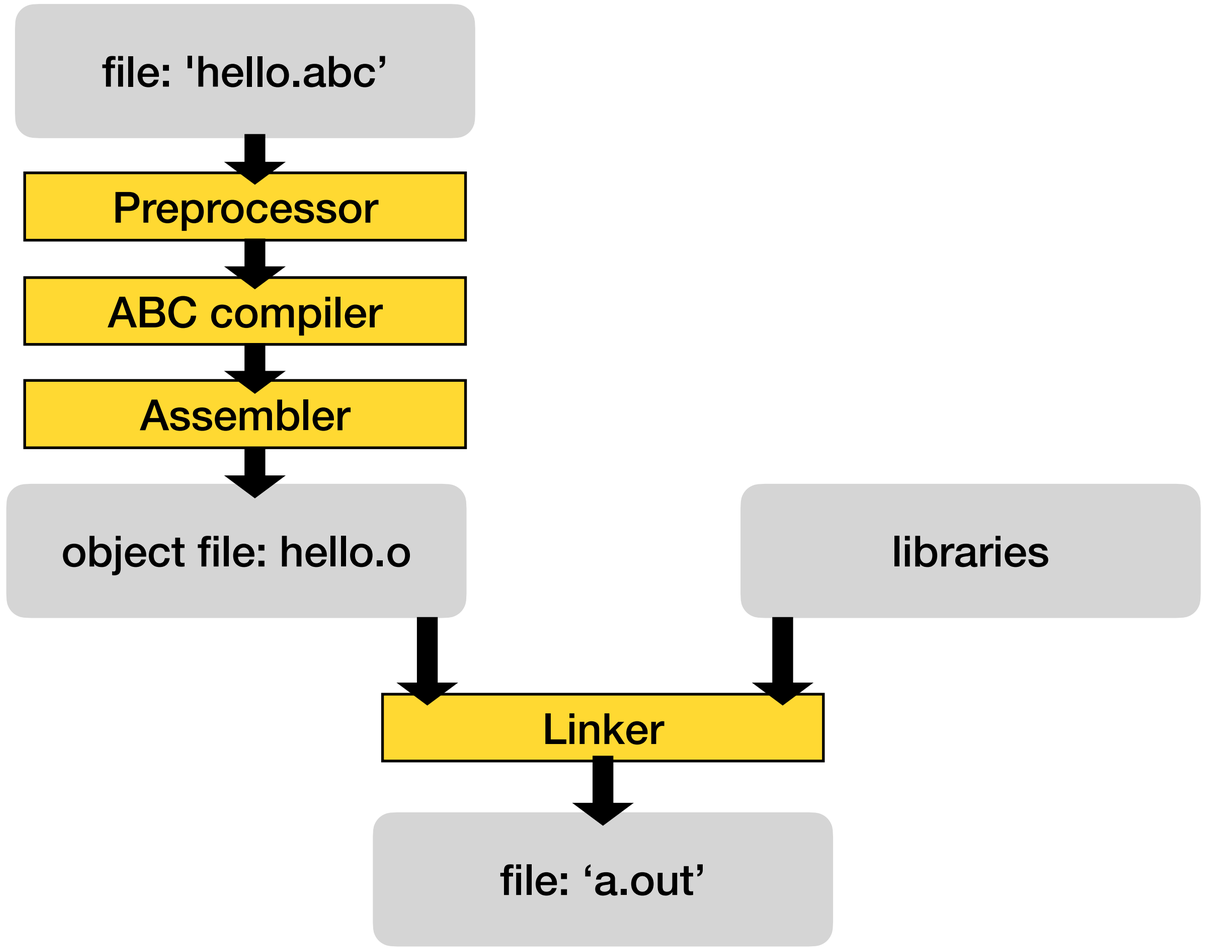


Phases of Translation

```
MCL:abc-examples lehn$ objdump -D hello.o

hello.o:file format mach-o 64-bit x86-64

Disassembly of section __TEXT,__text:

0000000000000000 <_main>:
       0: 50                            pushq%rax
       1: 48 8d 3d 00 00 00 00          leaq(%rip), %rdi
       8: 31 c0                         xorl%eax, %eax
       a: e8 00 00 00 00                callq0xf <_main+0xf>
       f: 58                            popq%rax
      10: c3                            retq

Disassembly of section __TEXT,__const:

0000000000000011 <_.L0>:
      11: 68 65 6c 6c 6f                pushq$1869376613
      16: 2c 20                         subb$32, %al
      18: 77 6f                         ja0x89 <_.L0+0x78>
      1a: 72 6c                         jb0x88 <_.L0+0x77>
      1c: 64 21 0a                      andl%ecx, %fs:(%rdx)
      1f: 00                            <unknown>
```



Phases of Translation



abc -S factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

abc -S factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .p2align        4, 0x90
_factorial:
        movl    %edi, -8(%rsp)
        movl    $1, -16(%rsp)
        movl    $1, -12(%rsp)
        .p2align        4, 0x90
LBB0_1:
        movl    -12(%rsp), %eax
        cmpl    -8(%rsp), %eax
        jg      LBB0_3
        movl    -12(%rsp), %eax
        movl    -16(%rsp), %ecx
        imull   %eax, %ecx
        movl    %ecx, -16(%rsp)
        incl    %eax
        movl    %eax, -12(%rsp)
        jmp     LBB0_1
LBB0_3:
        movl    -16(%rsp), %eax
        movl    %eax, -4(%rsp)
        movl    -4(%rsp), %eax
        retq

        .globl  _main
        .p2align        4, 0x90
_main:
        pushq   %rax
        movl    $5, %edi
        callq   _factorial
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    %eax, %edx
        xorl    %eax, %eax
        callq   _printf
        popq    %rax
        retq

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

abc -S -O1 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

abc -S -O1 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .globl  _main
        .p2align        4, 0x90
_main:
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    $120, %edx
        xorl    %eax, %eax
        jmp     _printf

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

abc -S -O1 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .globl  _main
        .p2align        4, 0x90
_main:
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    $120, %edx
        xorl    %eax, %eax
        jmp     _printf

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

```
@ <stdio.hdr>

fn main()
{
    printf("%d! = %d\n", 5, 120);
}
```

abc -S factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

abc -S factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .p2align        4, 0x90
_factorial:
        pushq   %rbx
        subq    $16, %rsp
        movl    %edi, 12(%rsp)
        cmpl    $2, %edi
        jl      LBB0_2
        movl    12(%rsp), %ebx
        leal    -1(%rbx), %edi
        callq   _factorial
        imull   %ebx, %eax
        movl    %eax, 8(%rsp)
        jmp     LBB0_3
LBB0_2:
        movl    $1, 8(%rsp)
LBB0_3:
        movl    8(%rsp), %eax
        addq    $16, %rsp
        popq    %rbx
        retq

        .globl  _main
        .p2align        4, 0x90
_main:
        pushq   %rax
        movl    $5, %edi
        callq   _factorial
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    %eax, %edx
        xorl    %eax, %eax
        callq   _printf
        popq    %rax
        retq

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

abc -S -O1 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

abc -S -O1 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .p2align        4, 0x90
_factorial:
        movl    $1, %eax
        cmpl    $2, %edi
        jl      LBB0_3
        .p2align        4, 0x90
LBB0_2:
        imull   %edi, %eax
        decl    %edi
        cmpl    $2, %edi
        jge     LBB0_2
LBB0_3:
        retq

        .globl  _main
        .p2align        4, 0x90
_main:
        pushq   %rax
        movl    $5, %edi
        callq   _factorial
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    %eax, %edx
        xorl    %eax, %eax
        popq    %rcx
        jmp     _printf

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

abc -S -O2 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

abc -S -O2 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .globl  _main
        .p2align        4, 0x90
_main:
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    $120, %edx
        xorl    %eax, %eax
        jmp     _printf

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

abc -S -O2 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    if (n > 1) {
        return n * factorial(n - 1);
    }
    return 1;
}

fn main()
{
    printf("%d! = %d\n", 5, factorial(5));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .globl  _main
        .p2align        4, 0x90
_main:
        leaq    _.L0(%rip), %rdi
        movl    $5, %esi
        movl    $120, %edx
        xorl    %eax, %eax
        jmp     _printf

        .section        __TEXT,__const
_.L0:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

```
@ <stdio.hdr>

fn main()
{
    printf("%d! = %d\n", 5, 120);
}
```

abc -S factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    local n: int;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("%d! = %d\n", n, factorial(n));
}
```

abc -S factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    local n: int;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("%d! = %d\n", n, factorial(n));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .p2align        4, 0x90
_factorial:
        movl    %edi, -8(%rsp)
        movl    $1, -16(%rsp)
        movl    $1, -12(%rsp)
        .p2align        4, 0x90
LBB0_1:
        movl    -12(%rsp), %eax
        cmpl    -8(%rsp), %eax
        jg      LBB0_3
        movl    -12(%rsp), %eax
        movl    -16(%rsp), %ecx
        imull   %eax, %ecx
        movl    %ecx, -16(%rsp)
        incl    %eax
        movl    %eax, -12(%rsp)
        jmp     LBB0_1
LBB0_3:
        movl    -16(%rsp), %eax
        movl    %eax, -4(%rsp)
        movl    -4(%rsp), %eax
        retq

        .globl  _main
        .p2align        4, 0x90
_main:
        pushq   %rbx
        subq    $16, %rsp
        leaq    _.L0(%rip), %rdi
        xorl    %eax, %eax
        callq   _printf
        leaq    _.L1(%rip), %rdi
        leaq    12(%rsp), %rsi
        xorl    %eax, %eax
        callq   _scanf
        movl    12(%rsp), %ebx
        movl    %ebx, %edi
        callq   _factorial
        leaq    _.L2(%rip), %rdi
        movl    %ebx, %esi
        movl    %eax, %edx
        xorl    %eax, %eax
        callq   _printf
        addq    $16, %rsp
        popq    %rbx
        retq

        .section        __TEXT,__const
_.L0:
        .asciz  "Enter n: "

_.L1:
        .asciz  "%d"

_.L2:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```

abc -S -O3 factorial.abc

```
@ <stdio.hdr>

fn factorial(n: int): int
{
    local result: int = 1;
    for (local i: int = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}

fn main()
{
    local n: int;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("%d! = %d\n", n, factorial(n));
}
```

```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 14, 0
        .globl  _main
        .p2align        4, 0x90
_main:
        pushq   %rax
        leaq    _.L0(%rip), %rdi
        xorl    %eax, %eax
        callq   _printf
        leaq    _.L1(%rip), %rdi
        leaq    4(%rsp), %rsi
        xorl    %eax, %eax
        callq   _scanf
        movl    4(%rsp), %esi
        testl   %esi, %esi
        jle     LBB0_1
        movl    $1, %eax
        movl    $1, %edx
        .p2align        4, 0x90
LBB0_3:
        imull   %eax, %edx
        incl    %eax
        cmpl    %esi, %eax
        jle     LBB0_3
        jmp     LBB0_4
LBB0_1:
        movl    $1, %edx
LBB0_4:
        leaq    _.L2(%rip), %rdi
        xorl    %eax, %eax
        callq   _printf
        popq    %rax
        retq

        .section        __TEXT,__const
_.L0:
        .asciz  "Enter n: "

_.L1:
        .asciz  "%d"

_.L2:
        .asciz  "%d! = %d\n"

.subsections_via_symbols
```