created: 2022-09-15
last-modified: "2022-09-19"
tags: study, university

2018130508 조건희

What is `Makefile`?

`Makefile` is used when code files are builded. For example, following is `Makefile` of native compilation of x86.

```
CC=gcc

all: compare

compare: compare.o
	objdump -SD -Mintel compare.o > compare.odump
#   make .odump file since object file is binary
#   can't link since .c file doesn't have main() func
#   in this case x86 compiler cannot make executable file

compare.o: compare.s
	as compare.s -o compare.o

compare.s: compare.i
	$(CC) -S compare.i

compare.i: compare.c
	cpp compare.c > compare.i

clean:
	rm -rf *.i *.s *.o *.mif *.dump *.odump *.hex *.bin *.mif labcode
```

First, I get `compare.i`. After that, I get `compare.s`. Then I get `compare.o`. Finally, I dump `.o` file into `compare.odump` in x86 readable code. `make clean` runs `rm` code which removes all generated file(in fact, I don't need to delete some files but I included it for no reason).
So, role of `Makefile` is automation of each steps of compilation(or build, or creating executable(or object) file).

Output of x86 Compilation

Preprocessing

```
# 1 "compare.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "compare.c"

int compare(int b, int c)
{
    int a;
    a = ((b) < (c) ? (b) : (c));;
    return a;
}
```

Compilation

```
        .file   "compare.c"
        .text
        .globl  compare
        .def    compare ;       .scl    2;      .type   32; .endef
        .seh_proc       compare
compare:
        pushq   %rbp
        .seh_pushreg    %rbp
        movq    %rsp, %rbp
        .seh_setframe   %rbp, 0
        subq    $   16, %rsp
        .seh_stackalloc     16
        .seh_endprologue
        movl    %ecx, 16(%rbp)
        movl    %edx, 24(%rbp)
        movl    16(%rbp), %eax
        cmpl    %eax, 24(%rbp)
        cmovle    24(%rbp), %eax
        movl    %eax, -4(%rbp)
        movl    -4(%rbp), %eax
        addq    $   16, %rsp
        popq    %rbp
        ret
        .seh_endproc
        .ident  "GCC: (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0"
```

Assembler

This is `objdump` output since object file is binary.

```
compare.o:     file format pe-x86-64


Disassembly of section .text:

0000000000000000 <compare>:
   0:   55                      push   rbp
   1:   48 89 e5                mov    rbp,rsp
   4:   48 83 ec 10             sub    rsp,0x10
   8:   89 4d 10                mov    DWORD PTR [rbp+0x10],ecx
   b:   89 55 18                mov    DWORD PTR [rbp+0x18],edx
   e:   8b 45 10                mov    eax,DWORD PTR [rbp+0x10]
  11:   39 45 18                cmp    DWORD PTR [rbp+0x18],eax
  14:   0f 4e 45 18             cmovle eax,DWORD PTR [rbp+0x18]
  18:   89 45 fc                mov    DWORD PTR [rbp-0x4],eax
  1b:   8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  1e:   48 83 c4 10             add    rsp,0x10
  22:   5d                      pop    rbp
  23:   c3                      ret
  24:   90                      nop
  25:   90                      nop
  26:   90                      nop
  27:   90                      nop
  28:   90                      nop
  29:   90                      nop
  2a:   90                      nop
  2b:   90                      nop
  2c:   90                      nop
  2d:   90                      nop
  2e:   90                      nop
  2f:   90                      nop

Disassembly of section .xdata:

0000000000000000 <.xdata>:
   0:   01 08                   add    DWORD PTR [rax],ecx
   2:   03 05 08 12 04 03       add    eax,DWORD PTR [rip+0x3041208]        # 3041210
<.xdata+0x3041210>
   8:   01 50 00                add    DWORD PTR [rax+0x0],edx
        ...

Disassembly of section .pdata:

0000000000000000 <.pdata>:
   0:   00 00                   add    BYTE PTR [rax],al
   2:   00 00                   add    BYTE PTR [rax],al
```

```
   4:   24 00                   and    al,0x0
   6:   00 00                   add    BYTE PTR [rax],al
   8:   00 00                   add    BYTE PTR [rax],al
        ...

Disassembly of section .rdata$zzz:

0000000000000000 <.rdata$zzz>:
   0:   47                      rex.RXB
   1:   43                      rex.XB
   2:   43 3a 20                rex.XB cmp spl,BYTE PTR [r8]
   5:   28 78 38                sub    BYTE PTR [rax+0x38],bh
   8:   36 5f                   ss pop rdi
   a:   36 34 2d                ss xor al,0x2d
   d:   70 6f                   jo     7e <.rdata$zzz+0x7e>
   f:   73 69                   jae    7a <.rdata$zzz+0x7a>
  11:   78 2d                   js     40 <.rdata$zzz+0x40>
  13:   73 65                   jae    7a <.rdata$zzz+0x7a>
  15:   68 2d 72 65 76          push   0x7665722d
  1a:   30 2c 20                xor    BYTE PTR [rax+riz*1],ch
  1d:   42 75 69                rex.X jne 89 <.rdata$zzz+0x89>
  20:   6c                      ins    BYTE PTR es:[rdi],dx
  21:   74 20                   je     43 <.rdata$zzz+0x43>
  23:   62                      (bad)
  24:   79 20                   jns    46 <.rdata$zzz+0x46>
  26:   4d 69 6e 47 57 2d 57    imul   r13,QWORD PTR [r14+0x47],0x36572d57
  2d:   36
  2e:   34 20                   xor    al,0x20
  30:   70 72                   jo     a4 <.rdata$zzz+0xa4>
  32:   6f                      outs   dx,DWORD PTR ds:[rsi]
  33:   6a 65                   push   0x65
  35:   63 74 29 20             movsxd esi,DWORD PTR [rcx+rbp*1+0x20]
  39:   38 2e                   cmp    BYTE PTR [rsi],ch
  3b:   31 2e                   xor    DWORD PTR [rsi],ebp
  3d:   30 00                   xor    BYTE PTR [rax],al
        ...
```

Output of RISC-V Compilation

Preprocessing

```
# 1 "compare.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "compare.c"
```

```
int compare(int b, int c)
{
    int a;
    a = ((b) < (c) ? (b) : (c));;
    return a;
}
```

Compilation

```
        .file   "compare.c"
        .option nopic
        .attribute arch, "rv32i2p0_m2p0_a2p0_c2p0"
        .attribute unaligned_access, 0
        .attribute stack_align, 16
        .text
        .align  1
        .globl  compare
        .type   compare, @function
compare:
        addi    sp, sp, -48
        sw      s0, 44(sp)
        addi    s0, sp, 48
        sw      a0, -36(s0)
        sw      a1, -40(s0)
        lw      a4, -36(s0)
        lw      a5, -40(s0)
        ble     a5, a4, .L2
        mv      a5, a4
.L2:
        sw      a5, -20(s0)
        lw      a5, -20(s0)
        mv      a0, a5
        lw      s0, 44(sp)
        addi    sp, sp, 48
        jr      ra
        .size   compare, .-compare
        .ident  "GCC: (GNU) 10.1.0"
```

Assembler

This is `objdump` output since object file is binary.

```
compare.o:     file format elf32-littleriscv
compare.o
```

```
architecture: riscv:rv32, flags 0x00000011:
HAS_RELOC, HAS_SYMS
start address 0x00000000

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000003c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000070  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000070  2**0
                  ALLOC
  3 .debug_info   0000006f  00000000  00000000  00000070  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  4 .debug_abbrev 0000005f  00000000  00000000  000000df  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_aranges 00000020  00000000  00000000  0000013e  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  6 .debug_line   00000052  00000000  00000000  0000015e  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
  7 .debug_str    0000006a  00000000  00000000  000001b0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  8 .comment      00000013  00000000  00000000  0000021a  2**0
                  CONTENTS, READONLY
  9 .debug_frame  00000034  00000000  00000000  00000230  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 10 .riscv.attributes 0000001c  00000000  00000000  00000264  2**0
                  CONTENTS, READONLY
SYMBOL TABLE:
00000000 l    df *ABS*  00000000 compare.c
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss   00000000 .bss
00000000 l       .text  00000000 .L0
00000000 l       .text  00000000 .L0
0000000c l       .text  00000000 .L0
00000014 l       .text  00000000 .L0
00000028 l       .text  00000000 .L0
0000002c l       .text  00000000 .L0
00000034 l       .text  00000000 .L0
0000003c l       .text  00000000 .L0
00000000 l    d  .debug_info    00000000 .debug_info
00000000 l    d  .debug_abbrev  00000000 .debug_abbrev
00000000 l    d  .debug_aranges 00000000 .debug_aranges
00000000 l    d  .debug_line    00000000 .debug_line
00000000 l    d  .debug_str     00000000 .debug_str
0000003c l       .text  00000000 .L0
00000000 l       .debug_frame   00000000 .L0
00000024 l       .text  00000000 .L2
```

```
00000000 l       .debug_abbrev  00000000 .Ldebug_abbrev0
00000000 l       .debug_str     00000000 .LASF0
00000058 l       .debug_str     00000000 .LASF1
00000039 l       .debug_str     00000000 .LASF2
00000000 l       .text  00000000 .Ltext0
0000003c l       .text  00000000 .Letext0
00000000 l       .debug_line    00000000 .Ldebug_line0
00000062 l       .debug_str     00000000 .LASF3
00000000 l       .text  00000000 .LFB0
0000003c l       .text  00000000 .LFE0
00000000 l       .debug_info    00000000 .Ldebug_info0
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .debug_frame   00000000 .debug_frame
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 g     F .text  0000003c compare



Disassembly of section .text:

00000000 <compare>:
#define min(x, y) ((x) < (y) ? (x) : (y));
int compare(int b, int c)
{
   0:   fd010113                addi    sp,sp,-48
   4:   02812623                sw      s0,44(sp)
   8:   03010413                addi    s0,sp,48
   c:   fca42e23                sw      a0,-36(s0)
  10:   fcb42c23                sw      a1,-40(s0)
    int a;
    a = min(b, c);
  14:   fdc42703                lw      a4,-36(s0)
  18:   fd842783                lw      a5,-40(s0)
  1c:   00f75463                bge     a4,a5,24 <.L2>
                        1c: R_RISCV_BRANCH      .L2
  20:   00070793                mv      a5,a4

00000024 <.L2>:
  24:   fef42623                sw      a5,-20(s0)
    return a;
  28:   fec42783                lw      a5,-20(s0)
  2c:   00078513                mv      a0,a5
  30:   02c12403                lw      s0,44(sp)
  34:   03010113                addi    sp,sp,48
  38:   00008067                ret
```

Linker

This is `objdump` output since object file is binary.

```
labcode:     file format elf32-littleriscv
labcode
architecture: riscv:rv32, flags 0x00000012:
EXEC_P, HAS_SYMS
start address 0x00000000

Program Header:
    LOAD off    0x00000060 vaddr 0x00000000 paddr 0x00000000 align 2**4
         filesz 0x00000800 memsz 0x00000800 flags rwx

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000004c  00000000  00000000  00000060  2**4
                  CONTENTS, ALLOC, LOAD, CODE
  1 .data         00000400  00000400  00000400  00000460  2**4
                  CONTENTS, ALLOC, LOAD, DATA
  2 .riscv.attributes 0000001c  00000000  00000000  00000860  2**0
                  CONTENTS, READONLY
  3 .comment      00000012  00000000  00000000  0000087c  2**0
                  CONTENTS, READONLY
  4 .debug_line   0000008d  00000000  00000000  0000088e  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  5 .debug_info   00000095  00000000  00000000  0000091b  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  6 .debug_abbrev 00000073  00000000  00000000  000009b0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  7 .debug_aranges 00000040  00000000  00000000  00000a28  2**3
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  8 .debug_str    0000007d  00000000  00000000  00000a68  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
  9 .debug_frame  00000034  00000000  00000000  00000ae8  2**2
                  CONTENTS, READONLY, DEBUGGING, OCTETS
SYMBOL TABLE:
00000000 l    d  .text  00000000 .text
00000400 l    d  .data  00000000 .data
00000000 l    d  .riscv.attributes      00000000 .riscv.attributes
00000000 l    d  .comment       00000000 .comment
00000000 l    d  .debug_line    00000000 .debug_line
00000000 l    d  .debug_info    00000000 .debug_info
00000000 l    d  .debug_abbrev  00000000 .debug_abbrev
00000000 l    d  .debug_aranges 00000000 .debug_aranges
00000000 l    d  .debug_str     00000000 .debug_str
00000000 l    d  .debug_frame   00000000 .debug_frame
00000000 l    df *ABS*  00000000 lab0.o
00000400 l       .data  00000000 stack
00000000 l    df *ABS*  00000000 compare.c
00000010 g     F .text  0000003c compare
```

```
Disassembly of section .text:

00000000 <compare-0x10>:
.text
.align 4

        la sp, stack
   0:   40000113                li      sp,1024
        j       compare
   4:   00c0006f                j       10 <compare>
        ...

00000010 <compare>:
#define min(x, y) ((x) < (y) ? (x) : (y));
int compare(int b, int c)
{
  10:   fd010113                addi    sp,sp,-48
  14:   02812623                sw      s0,44(sp)
  18:   03010413                addi    s0,sp,48
  1c:   fca42e23                sw      a0,-36(s0)
  20:   fcb42c23                sw      a1,-40(s0)
    int a;
    a = min(b, c);
  24:   fdc42703                lw      a4,-36(s0)
  28:   fd842783                lw      a5,-40(s0)
  2c:   00f75463                bge     a4,a5,34 <compare+0x24>
  30:   00070793                mv      a5,a4
  34:   fef42623                sw      a5,-20(s0)
    return a;
  38:   fec42783                lw      a5,-20(s0)
  3c:   00078513                mv      a0,a5
  40:   02c12403                lw      s0,44(sp)
  44:   03010113                addi    sp,sp,48
  48:   00008067                ret
```