```
walterzh$ arm-linux-gnueabi-ld -verbose
GNU ld (GNU Binutils for Ubuntu) 2.22
  Supported emulations:
   armelf_linux_eabi
   armelfb_linux_eabi
using internal linker script:
==================================================
/* Script for -z combreloc: combine and sort reloc sections */
OUTPUT_FORMAT("elf32-littlearm", "elf32-bigarm",
              "elf32-littlearm")
OUTPUT_ARCH(arm)
ENTRY(_start)
SEARCH_DIR("/lib/arm-linux-gnueabi"); SEARCH_DIR("/usr/lib/arm-linux-
gnueabi");SEARCH_DIR("/usr/arm-linux-gnueabi/lib");
SECTIONS
{
  /* Read-only sections, merged into text segment: */
  PROVIDE (__executable_start = SEGMENT_START("text-segment", 0x00008000)); . =
SEGMENT_START("text-segment", 0x00008000) + SIZEOF_HEADERS;
  .interp         : { *(.interp) }
  .note.gnu.build-id : { *(.note.gnu.build-id) }
  .hash           : { *(.hash) }
  .gnu.hash       : { *(.gnu.hash) }
  .dynsym         : { *(.dynsym) }
  .dynstr         : { *(.dynstr) }
```

```
.gnu.version    : { *(.gnu.version) }
.gnu.version_d  : { *(.gnu.version_d) }
.gnu.version_r  : { *(.gnu.version_r) }
.rel.dyn        :
 {
   *(.rel.init)
   *(.rel.text .rel.text.* .rel.gnu.linkonce.t.*)
   *(.rel.fini)
   *(.rel.rodata .rel.rodata.* .rel.gnu.linkonce.r.*)
   *(.rel.data.rel.ro* .rel.gnu.linkonce.d.rel.ro.*)
   *(.rel.data .rel.data.* .rel.gnu.linkonce.d.*)
   *(.rel.tdata .rel.tdata.* .rel.gnu.linkonce.td.*)
   *(.rel.tbss .rel.tbss.* .rel.gnu.linkonce.tb.*)
   *(.rel.ctors)
   *(.rel.dtors)
   *(.rel.got)
   *(.rel.bss .rel.bss.* .rel.gnu.linkonce.b.*)
   PROVIDE_HIDDEN (__rel_iplt_start = .);
   *(.rel.iplt)
   PROVIDE_HIDDEN (__rel_iplt_end = .);
   PROVIDE_HIDDEN (__rela_iplt_start = .);
   PROVIDE_HIDDEN (__rela_iplt_end = .);
 }
.rela.dyn       :
 {
```

```
    *(.rela.init)
    *(.rela.text .rela.text.* .rela.gnu.linkonce.t.*)
    *(.rela.fini)
    *(.rela.rodata .rela.rodata.* .rela.gnu.linkonce.r.*)
    *(.rela.data .rela.data.* .rela.gnu.linkonce.d.*)
    *(.rela.tdata .rela.tdata.* .rela.gnu.linkonce.td.*)
    *(.rela.tbss .rela.tbss.* .rela.gnu.linkonce.tb.*)
    *(.rela.ctors)
    *(.rela.dtors)
    *(.rela.got)
    *(.rela.bss .rela.bss.* .rela.gnu.linkonce.b.*)
    PROVIDE_HIDDEN (__rel_iplt_start = .);
    PROVIDE_HIDDEN (__rel_iplt_end = .);
    PROVIDE_HIDDEN (__rela_iplt_start = .);
    *(.rela.iplt)
    PROVIDE_HIDDEN (__rela_iplt_end = .);
  }
.rel.plt        :
  {
    *(.rel.plt)
  }
.rela.plt       :
  {
    *(.rela.plt)
  }
```

```
.init           :
{
  KEEP (*(.init))
} =0
.plt            : { *(.plt) }
.iplt           : { *(.iplt) }
.text           :
{
  *(.text.unlikely .text.*_unlikely)
  *(.text.exit .text.exit.*)
  *(.text.startup .text.startup.*)
  *(.text.hot .text.hot.*)
  *(.text .stub .text.* .gnu.linkonce.t.*)
  /* .gnu.warning sections are handled specially by elf32.em.  */
  *(.gnu.warning)
  *(.glue_7t) *(.glue_7) *(.vfp11_veneer) *(.v4_bx)
} =0
.fini           :
{
  KEEP (*(.fini))
} =0
PROVIDE (__etext = .);
PROVIDE (_etext = .);
PROVIDE (etext = .);
.rodata         : { *(.rodata .rodata.* .gnu.linkonce.r.*) }
```

```
  .rodata1        : { *(.rodata1) }

  .ARM.extab   : { *(.ARM.extab* .gnu.linkonce.armextab.*) }

   PROVIDE_HIDDEN (__exidx_start = .);

  .ARM.exidx   : { *(.ARM.exidx* .gnu.linkonce.armexidx.*) }

   PROVIDE_HIDDEN (__exidx_end = .);

  .eh_frame_hdr : { *(.eh_frame_hdr) }

  .eh_frame       : ONLY_IF_RO { KEEP (*(.eh_frame)) }

  .gcc_except_table   : ONLY_IF_RO { *(.gcc_except_table

  .gcc_except_table.*) }

  /* These sections are generated by the Sun/Oracle C++ compiler.  */

  .exception_ranges   : ONLY_IF_RO { *(.exception_ranges

  .exception_ranges*) }

  /* Adjust the address for the data segment.  We want to adjust up to

     the same address within the page on the next page up.  */

  . = ALIGN (CONSTANT (MAXPAGESIZE)) - ((CONSTANT (MAXPAGESIZE) - .) & (CONSTANT
(MAXPAGESIZE) - 1)); . = DATA_SEGMENT_ALIGN (CONSTANT (MAXPAGESIZE), CONSTANT
(COMMONPAGESIZE));

  /* Exception handling  */

  .eh_frame       : ONLY_IF_RW { KEEP (*(.eh_frame)) }

  .gcc_except_table   : ONLY_IF_RW { *(.gcc_except_table .gcc_except_table.*) }

  .exception_ranges   : ONLY_IF_RW { *(.exception_ranges .exception_ranges*) }

  /* Thread Local Storage sections  */

  .tdata       : { *(.tdata .tdata.* .gnu.linkonce.td.*) }

  .tbss              : { *(.tbss .tbss.* .gnu.linkonce.tb.*) *(.tcommon) }

  .preinit_array     :

  {
```

```
  PROVIDE_HIDDEN (__preinit_array_start = .);
  KEEP (*(.preinit_array))
  PROVIDE_HIDDEN (__preinit_array_end = .);
}
.init_array     :
{
  PROVIDE_HIDDEN (__init_array_start = .);
  KEEP (*(SORT(.init_array.*)))
  KEEP (*(.init_array))
  PROVIDE_HIDDEN (__init_array_end = .);
}
.fini_array     :
{
  PROVIDE_HIDDEN (__fini_array_start = .);
  KEEP (*(SORT(.fini_array.*)))
  KEEP (*(.fini_array))
  PROVIDE_HIDDEN (__fini_array_end = .);
}
.ctors          :
{
  /* gcc uses crtbegin.o to find the start of
     the constructors, so we make sure it is
     first.  Because this is a wildcard, it
     doesn't matter if the user does not
     actually link against crtbegin.o; the
```

```
    linker won't look for a file to match a
    wildcard.  The wildcard also means that it
    doesn't matter which directory crtbegin.o
    is in.  */
  KEEP (*crtbegin.o(.ctors))
  KEEP (*crtbegin?.o(.ctors))
  /* We don't want to include the .ctor section from
    the crtend.o file until after the sorted ctors.
    The .ctor section from the crtend file contains the
    end of ctors marker and it must be last */
  KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o ) .ctors))
  KEEP (*(SORT(.ctors.*)))
  KEEP (*(.ctors))
 }
 .dtors          :
 {
  KEEP (*crtbegin.o(.dtors))
  KEEP (*crtbegin?.o(.dtors))
  KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o ) .dtors))
  KEEP (*(SORT(.dtors.*)))
  KEEP (*(.dtors))
 }
 .jcr            : { KEEP (*(.jcr)) }
 .data.rel.ro : { *(.data.rel.ro.local* .gnu.linkonce.d.rel.ro.local.*) *(.data.rel.ro*
.gnu.linkonce.d.rel.ro.*) }
```

```
.dynamic        : { *(.dynamic) }
. = DATA_SEGMENT_RELRO_END (0, .);
.got            : { *(.got.plt) *(.igot.plt) *(.got) *(.igot) }
.data           :
{
  PROVIDE (__data_start = .);
  *(.data .data.* .gnu.linkonce.d.*)
  SORT(CONSTRUCTORS)
}
.data1          : { *(.data1) }
_edata = .; PROVIDE (edata = .);
__bss_start = .;
__bss_start__ = .;
.bss            :
{
 *(.dynbss)
 *(.bss .bss.* .gnu.linkonce.b.*)
 *(COMMON)
 /* Align here to ensure that the .bss section occupies space up to
    _end.  Align after .bss to ensure correct alignment even if the
    .bss section disappears because there are no input sections.
    FIXME: Why do we need it? When there is no .bss section, we don't
    pad the .data section.  */
 . = ALIGN(. != 0 ? 32 / 8 : 1);
}
```

```
_bss_end__ = . ; __bss_end__ = . ;
. = ALIGN(32 / 8);
. = ALIGN(32 / 8);
__end__ = . ;
_end = .; PROVIDE (end = .);
. = DATA_SEGMENT_END (.);
/* Stabs debugging sections.  */
.stab          0 : { *(.stab) }
.stabstr       0 : { *(.stabstr) }
.stab.excl     0 : { *(.stab.excl) }
.stab.exclstr  0 : { *(.stab.exclstr) }
.stab.index    0 : { *(.stab.index) }
.stab.indexstr 0 : { *(.stab.indexstr) }
.comment       0 : { *(.comment) }
/* DWARF debug sections.
   Symbols in the DWARF debugging sections are relative to the beginning
   of the section so we begin them at 0.  */
/* DWARF 1 */
.debug          0 : { *(.debug) }
.line           0 : { *(.line) }
/* GNU DWARF 1 extensions */
.debug_srcinfo  0 : { *(.debug_srcinfo) }
.debug_sfnames  0 : { *(.debug_sfnames) }
/* DWARF 1.1 and DWARF 2 */
.debug_aranges  0 : { *(.debug_aranges) }
```

```
  .debug_pubnames 0 : { *(.debug_pubnames) }
  /* DWARF 2 */
  .debug_info     0 : { *(.debug_info .gnu.linkonce.wi.*) }
  .debug_abbrev   0 : { *(.debug_abbrev) }
  .debug_line     0 : { *(.debug_line) }
  .debug_frame    0 : { *(.debug_frame) }
  .debug_str      0 : { *(.debug_str) }
  .debug_loc      0 : { *(.debug_loc) }
  .debug_macinfo  0 : { *(.debug_macinfo) }
  /* SGI/MIPS DWARF 2 extensions */
  .debug_weaknames 0 : { *(.debug_weaknames) }
  .debug_funcnames 0 : { *(.debug_funcnames) }
  .debug_typenames 0 : { *(.debug_typenames) }
  .debug_varnames  0 : { *(.debug_varnames) }
  /* DWARF 3 */
  .debug_pubtypes 0 : { *(.debug_pubtypes) }
  .debug_ranges   0 : { *(.debug_ranges) }
  .gnu.attributes 0 : { KEEP (*(.gnu.attributes)) }
  .note.gnu.arm.ident 0 : { KEEP (*(.note.gnu.arm.ident)) }
  /DISCARD/ : { *(.note.GNU-stack) *(.gnu_debuglink) *(.gnu.lto_*) }
}
```

==========

```
root@granite2:~# ps  ./test_arm &
[1] 1719
469396fc 000083de
root@granite2:~# ps aux | grep test_arm
 1719 root      1544 S    ./test_arm
 1723 root      2248 S    grep test_arm
root@granite2:~# cat /proc/1719/maps
00008000-00009000 r-xp 00000000 b3:22 36732      /home/root/test_arm
00010000-00011000 r--p 00000000 b3:22 36732      /home/root/test_arm
00011000-00012000 rw-p 00001000 b3:22 36732      /home/root/test_arm
468f0000-4690f000 r-xp 00000000 b3:22 5895       /lib/ld-2.18.so
46916000-46917000 r--p 0001e000 b3:22 5895       /lib/ld-2.18.so
46917000-46918000 rw-p 0001f000 b3:22 5895       /lib/ld-2.18.so
46920000-46a4b000 r-xp 00000000 b3:22 5558       /lib/libc-2.18.so
46a4b000-46a52000 ---p 0012b000 b3:22 5558       /lib/libc-2.18.so
46a52000-46a54000 r--p 0012a000 b3:22 5558       /lib/libc-2.18.so
46a54000-46a56000 rw-p 0012c000 b3:22 5558       /lib/libc-2.18.so
46a56000-46a58000 rw-p 00000000 00:00 0
b6f69000-b6f6a000 rw-p 00000000 00:00 0
b6f6e000-b6f70000 rw-p 00000000 00:00 0
bea84000-beaa5000 rw-p 00000000 00:00 0          [stack]
becb2000-becb3000 r-xp 00000000 00:00 0          [sigpage]
ffff0000-ffff1000 r-xp 00000000 00:00 0          [vectors]
```

在embedded Linux上代码段从0x00008000开始。

原因如下：

PROVIDE (__executable_start = SEGMENT_START("text-segment", 0x00008000)); . = SEGMENT_START("text-segment", 0x00008000) + SIZEOF_HEADERS;

P.S.

```
walterzh$ cat test.c
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
	unsigned int lr_register = 100;
	unsigned int pc_register = 200;

	int ret_val = 5;

	asm (
		"mov %0, lr\n"
		"mov %1, pc\n"
		: "=r" (lr_register), "=r" (pc_register)
		:
		:
	);
```

```
	printf("%08x %08x\n", lr_register, pc_register);


	asm (
		"mov r0, %0\n"
		:
		: "r" (ret_val)
		:
	);


	while(1)
	{
		sleep(300);
	}

//	return ret_val;
}


arm-linux-gnueabi-gcc -g -fverbose-asm  test.c -o test_arm
```