Create a binary file and debugging

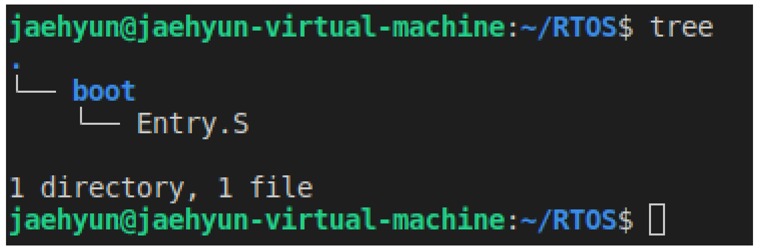
OPERATING SYSTEM BASED ON PBL

University of Ulsan
School of Electrical Engineering
Jaehyun Park
jaehyun@ulsan.ac.kr

Create a Skeleton Code

- Create ‘boot’ directory under the RTOS directory
- Create ‘Entry.S’ file at the boot directory

```
~/RTOS$ mkdir boot
~/RTOS$ cd boot
~/RTOS/boot$ vim Entry.S    #Any editor is fine
```



```
jaehyun@jaehyun-virtual-machine:~/RTOS$ tree
.
└── boot
    └── Entry.S

1 directory, 1 file
jaehyun@jaehyun-virtual-machine:~/RTOS$ 
```

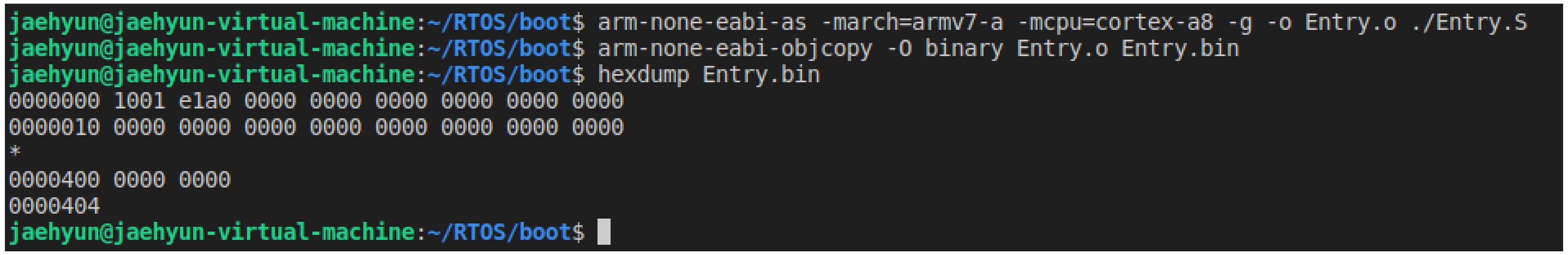
- Entry.S

```
.text
    .code 32

    .global vector_start
    .global vector_end

    vector_start:
        mov R0, R1
    vector_end:
        .space 1024, 0
.end
```

Compile the Code

- Use a cross toolchain to check the machine code

```
~/RTOS/boot$ arm-none-eabi-as -march=armv7-a -mcpu=cortex-a8 -g -o Entry.o ./Entry.S
~/RTOS/boot$ arm-none-eabi-objcopy -O binary Entry.o Entry.bin
~/RTOS/boot$ hexdump Entry.bin
```

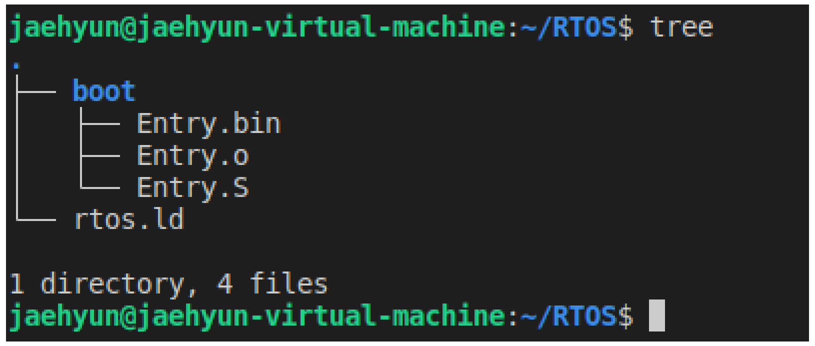


```
jaehyun@jaehyun-virtual-machine:~/RTOS/boot$ arm-none-eabi-as -march=armv7-a -mcpu=cortex-a8 -g -o Entry.o ./Entry.S
jaehyun@jaehyun-virtual-machine:~/RTOS/boot$ arm-none-eabi-objcopy -O binary Entry.o Entry.bin
jaehyun@jaehyun-virtual-machine:~/RTOS/boot$ hexdump Entry.bin
0000000 1001 e1a0 0000 0000 0000 0000 0000 0000
0000010 0000 0000 0000 0000 0000 0000 0000 0000
*
0000400 0000 0000
0000404
jaehyun@jaehyun-virtual-machine:~/RTOS/boot$
```

Create a Linker Script

- Create ‘rtos.ld’ at the RTOS directory
  - rtos.ld

```
ENTRY(vector_start)
SECTIONS
{
    . = 0x0;

    .text :
    {
        *(vector_start)
        *(.text .rodata)
    }
    .data :
    {
        *(.data)
    }
    .bss :
    {
        *(.bss)
    }
}
```



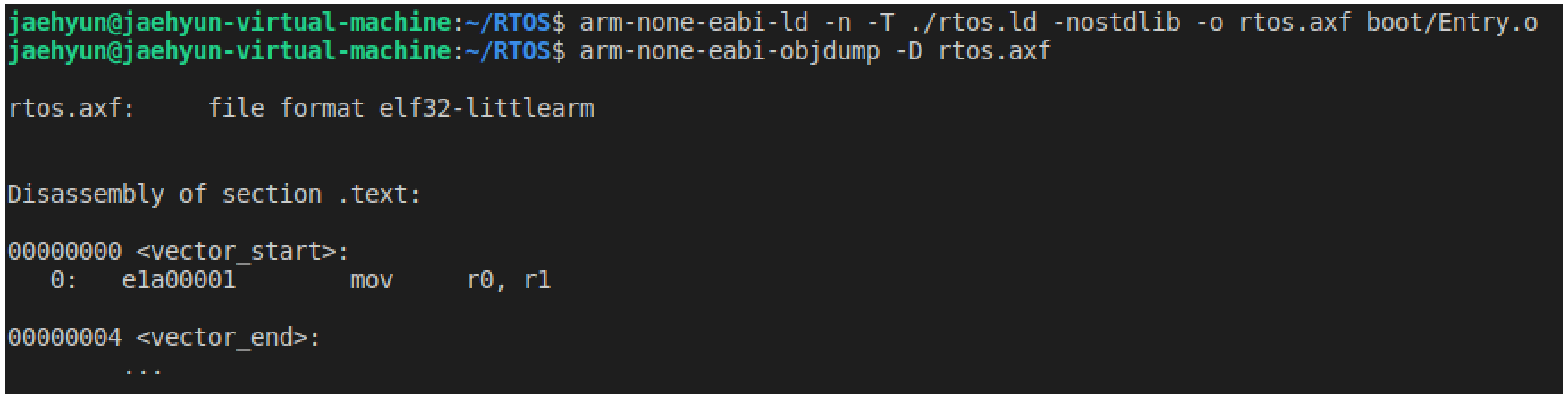
```
jaehyun@jaehyun-virtual-machine:~/RTOS$ tree
.
├── boot
│   ├── Entry.bin
│   ├── Entry.o
│   └── Entry.S
└── rtos.ld

1 directory, 4 files
jaehyun@jaehyun-virtual-machine:~/RTOS$ 
```

Create an Executable File

- Use a cross toolchain to create an executable file

Do not page align data   Read linker script   Only use library directories specified on the command line

```
~/RTOS$ arm-none-eabi-ld -n -T ./rtos.ld -nostdlib -o rtos.axf boot/Entry.o
~/RTOS$ arm-none-eabi-objdump -D rtos.axf
```

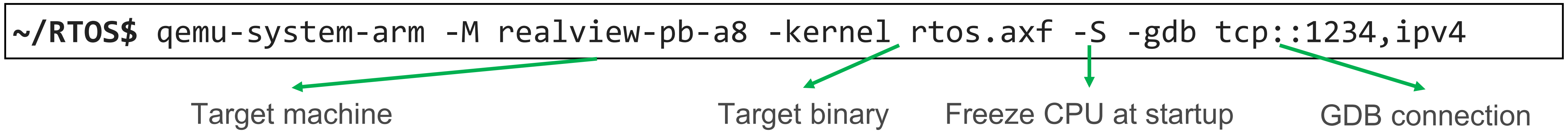


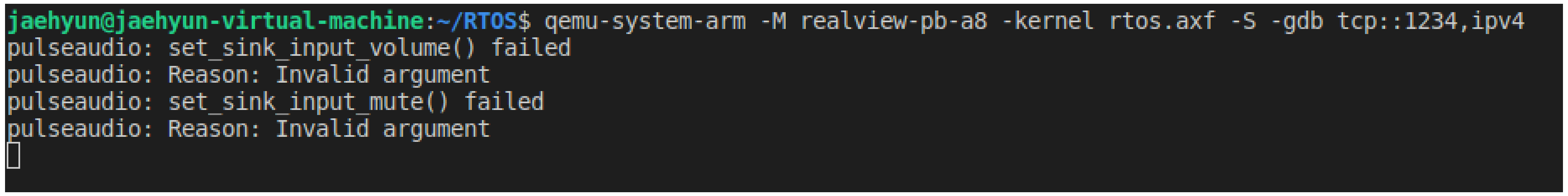
```
jaehyun@jaehyun-virtual-machine:~/RTOS$ arm-none-eabi-ld -n -T ./rtos.ld -nostdlib -o rtos.axf boot/Entry.o
jaehyun@jaehyun-virtual-machine:~/RTOS$ arm-none-eabi-objdump -D rtos.axf

rtos.axf:     file format elf32-littlearm


Disassembly of section .text:

00000000 <vector_start>:
   0:   e1a00001        mov     r0, r1

00000004 <vector_end>:
        ...
```

Run an Executable File

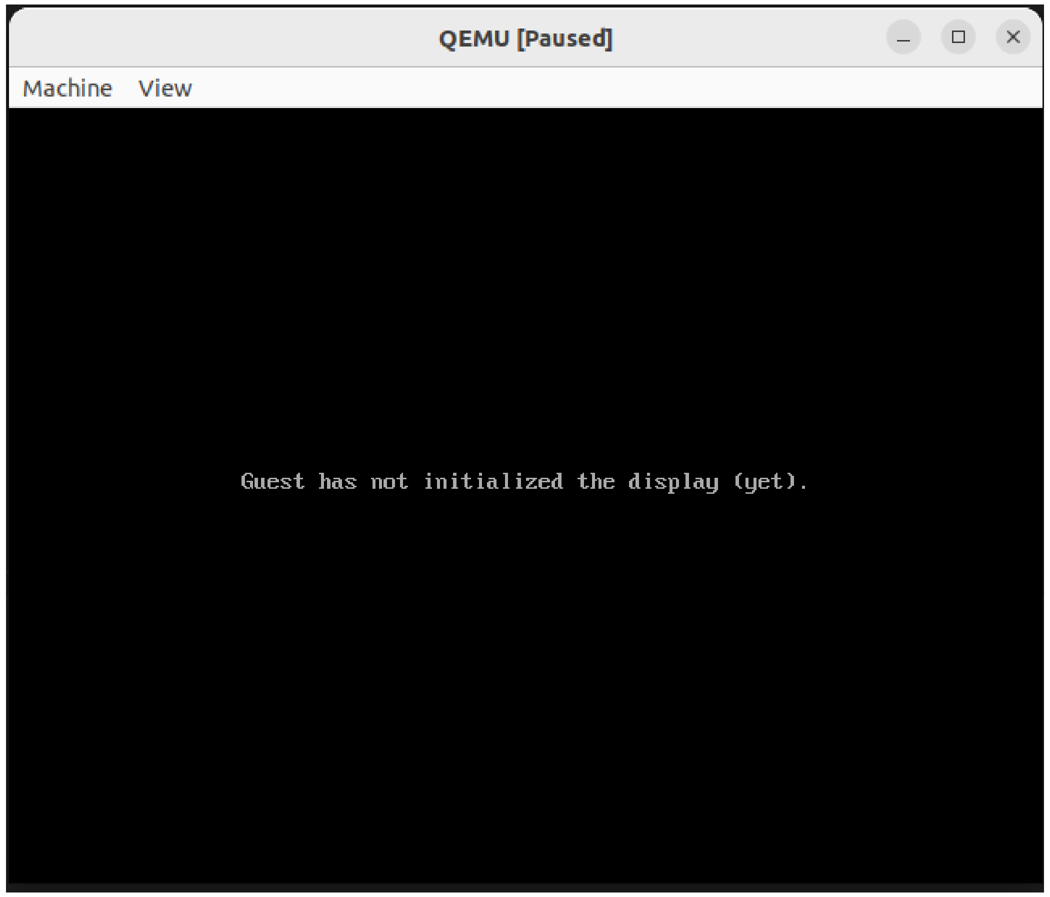
- Use QEMU to run the executable file

```
~/RTOS$ qemu-system-arm -M realview-pb-a8 -kernel rtos.axf -S -gdb tcp::1234,ipv4
```





```
jaehyun@jaehyun-virtual-machine:~/RTOS$ qemu-system-arm -M realview-pb-a8 -kernel rtos.axf -S -gdb tcp::1234,ipv4
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
```



Guest has not initialized the display (yet).

Debugging an Executable File

- Install GDB for ARM processor

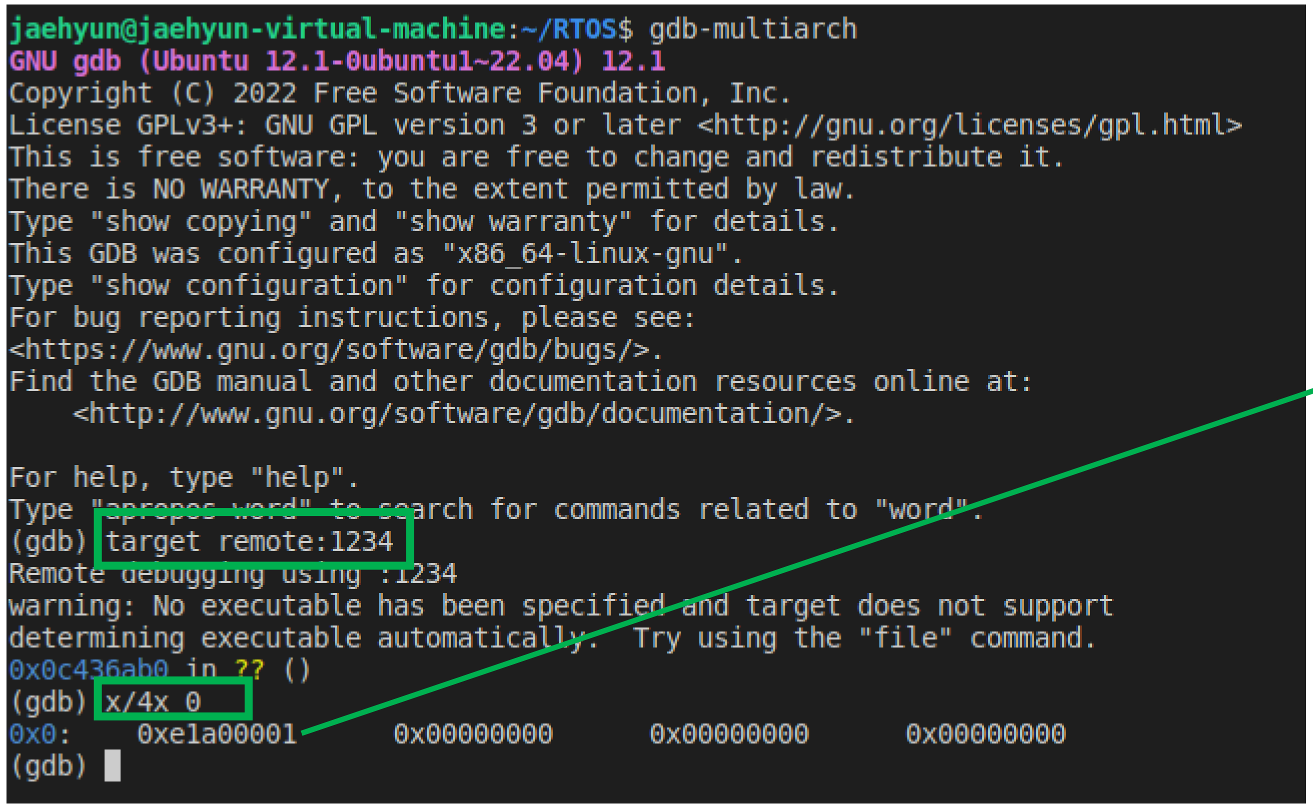
```
~/RTOS$ sudo apt install gdb-multiarch
```

- Run GDB and connect a debugging session

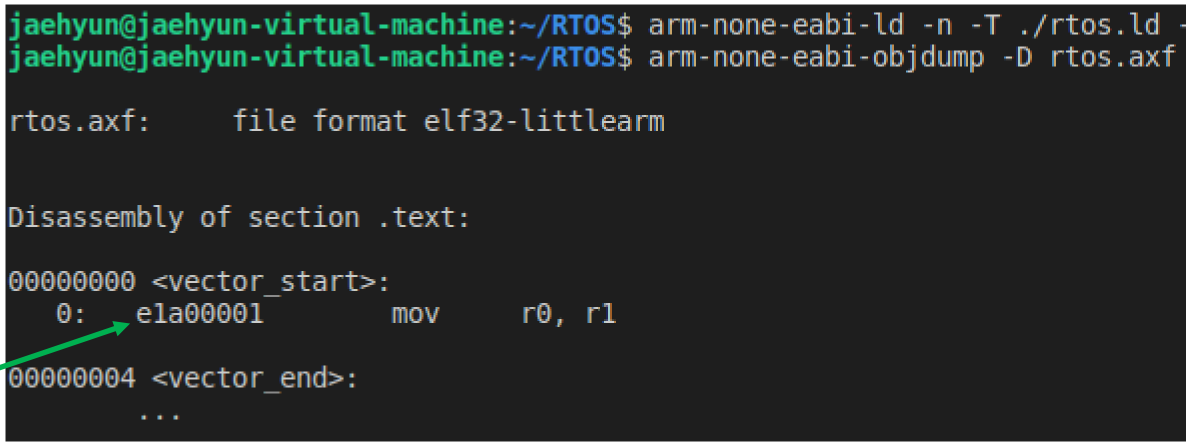
```
~/RTOS$ gdb-multiarch
```

  - Connect a debugging session

```
(GDB) target remote:1234
```



```
jaehyun@jaehyun-virtual-machine:~/RTOS$ gdb-multiarch
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) target remote:1234
Remote debugging using :1234
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x0c436ab0 in ?? ()
(gdb) x/4x 0
0x0:    0xe1a00001      0x00000000      0x00000000      0x00000000
(gdb) 
```

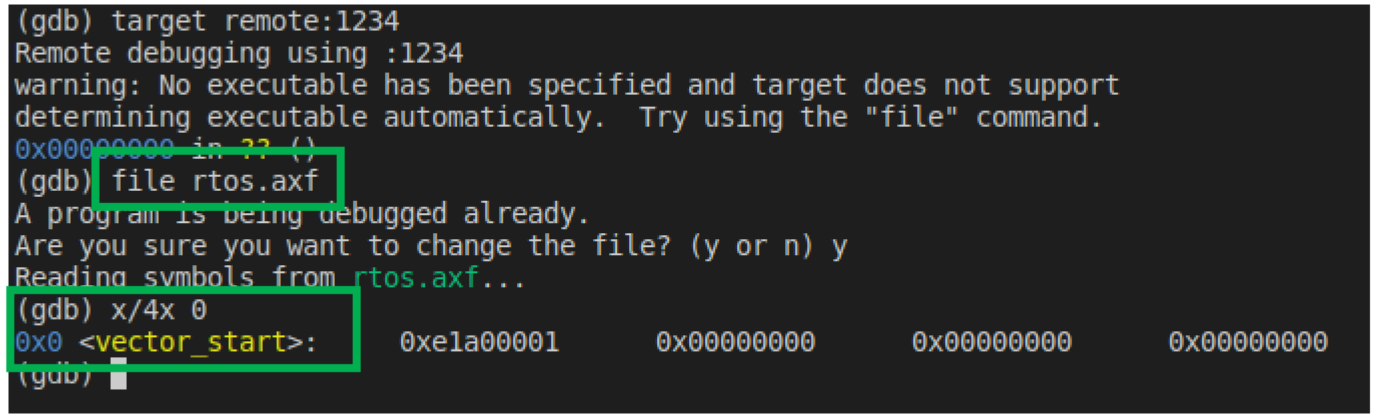


```
jaehyun@jaehyun-virtual-machine:~/RTOS$ arm-none-eabi-ld -n -T ./rtos.ld -
jaehyun@jaehyun-virtual-machine:~/RTOS$ arm-none-eabi-objdump -D rtos.axf

rtos.axf:     file format elf32-littlearm


Disassembly of section .text:

00000000 <vector_start>:
   0:   e1a00001        mov     r0, r1

00000004 <vector_end>:
        ...
```

Debugging an Executable File

- Load debugging symbol

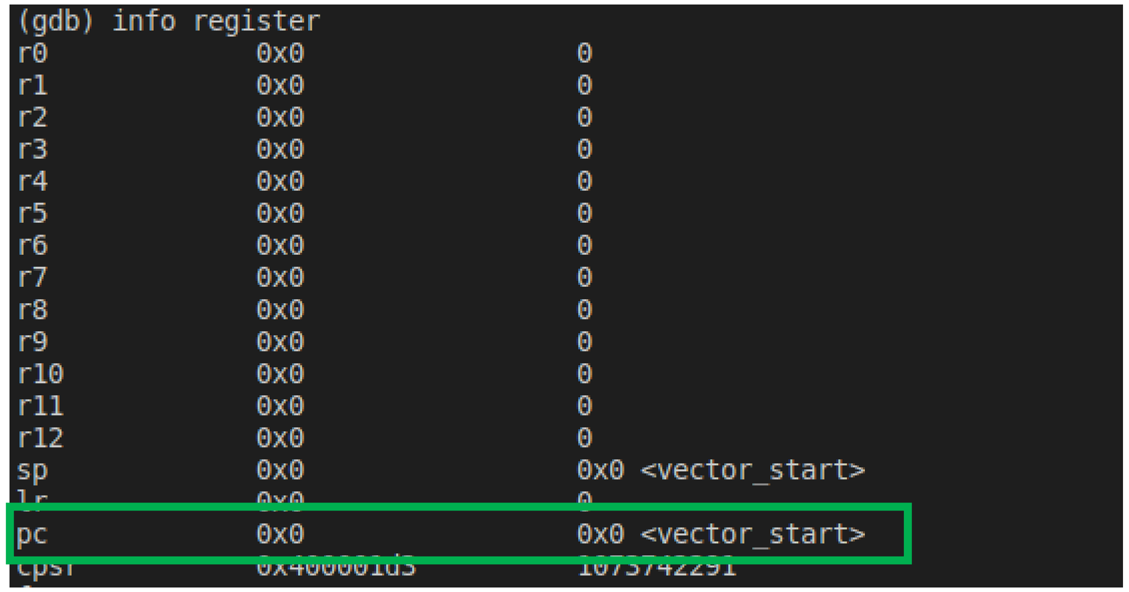
```
(GDB) file rtos.axf
```



```
(gdb) target remote:1234
Remote debugging using :1234
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x00000000 in ?? ()
(gdb) file rtos.axf
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from rtos.axf...
(gdb) x/4x 0
0x0 <vector_start>:     0xe1a00001      0x00000000      0x00000000      0x00000000
(gdb)
```

- List of integer registers and their contents

```
(GDB) info register
```

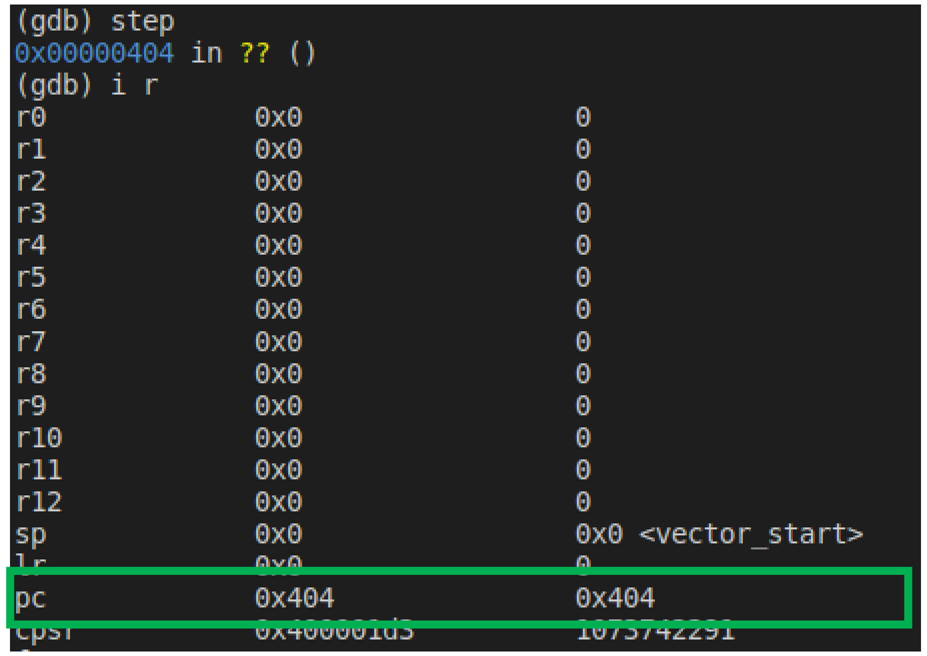


```
(gdb) info register
r0             0x0                 0
r1             0x0                 0
r2             0x0                 0
r3             0x0                 0
r4             0x0                 0
r5             0x0                 0
r6             0x0                 0
r7             0x0                 0
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
sp             0x0                 0x0 <vector_start>
lr             0x0                 0
pc             0x0                 0x0 <vector_start>
cpsr           0x400001d3          1073742291
```

Debugging an Executable File

- Step program until it reaches a different source line

```
(GDB) step
```

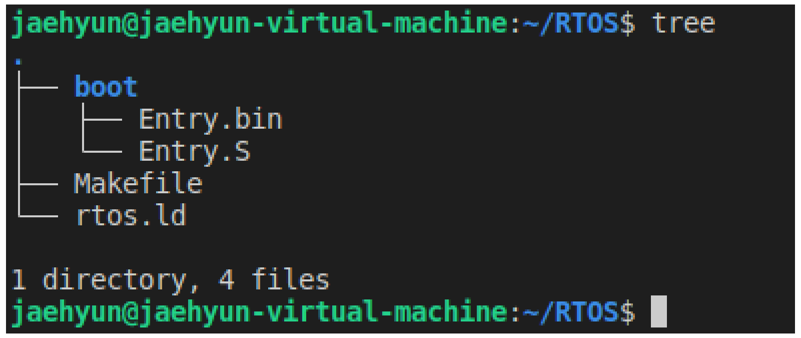


```
(gdb) step
0x00000404 in ?? ()
(gdb) i r
r0             0x0                 0
r1             0x0                 0
r2             0x0                 0
r3             0x0                 0
r4             0x0                 0
r5             0x0                 0
r6             0x0                 0
r7             0x0                 0
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
sp             0x0                 0x0 <vector_start>
lr             0x0                 0
pc             0x404               0x404
cpsr           0x400001d3          1073742291
```

Create a Makefile

- Makefile

```
ARCH = armv7-a
MCPU = cortex-a8

CC = arm-none-eabi-gcc
AS = arm-none-eabi-as
LD = arm-none-eabi-ld
OC = arm-none-eabi-objcopy

LINKER_SCRIPT = ./rtos.ld

ASM_SRCS = $(wildcard boot/*.S)
ASM_OBJS = $(patsubst boot/%.S, build/%.o, $(ASM_SRCS))

LDFLAGS = -nostdlib

rtos = build/rtos.axf
rtos_bin = build/rtos.bin

.PHONY: all clean run debug gdb

all: $(rtos)

clean:
    @rm -rf build

run:
    qemu-system-arm -M realview-pb-a8 -kernel $(rtos)

debug: $(rtos)
    qemu-system-arm -M realview-pb-a8 -kernel $(rtos) -S -gdb tcp::1234,ipv4

gdb:
    gdb-multiarch

$(rtos): $(ASM_OBJS) $(LINKER_SCRIPT)
    $(LD) -n -T $(LINKER_SCRIPT) -o $(rtos) $(ASM_OBJS)
    $(OC) -O binary $(rtos) $(rtos_bin)

build/%.o: boot/%.S
    mkdir -p $(shell dirname $@)
    $(AS) -march=$(ARCH) -mcpu=$(MCPU) -g -o $@ $<
```



```
jaehyun@jaehyun-virtual-machine:~/RTOS$ tree
.
├── boot
│   ├── Entry.bin
│   └── Entry.S
├── Makefile
└── rtos.ld

1 directory, 4 files
jaehyun@jaehyun-virtual-machine:~/RTOS$
```