

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DISCIPLINA: LABORATÓRIO DE PROCESSADORES- PCS3732
1º QUADRIMESTRE/2021



Aula 9
08 de Julho de 2021

GRUPO 10

Arthur Pires da Fonseca
Bruno José Mório
Iago Soriano Roque Monteiro

NUSP: 10773096
NUSP: 10336852
NUSP: 8572921

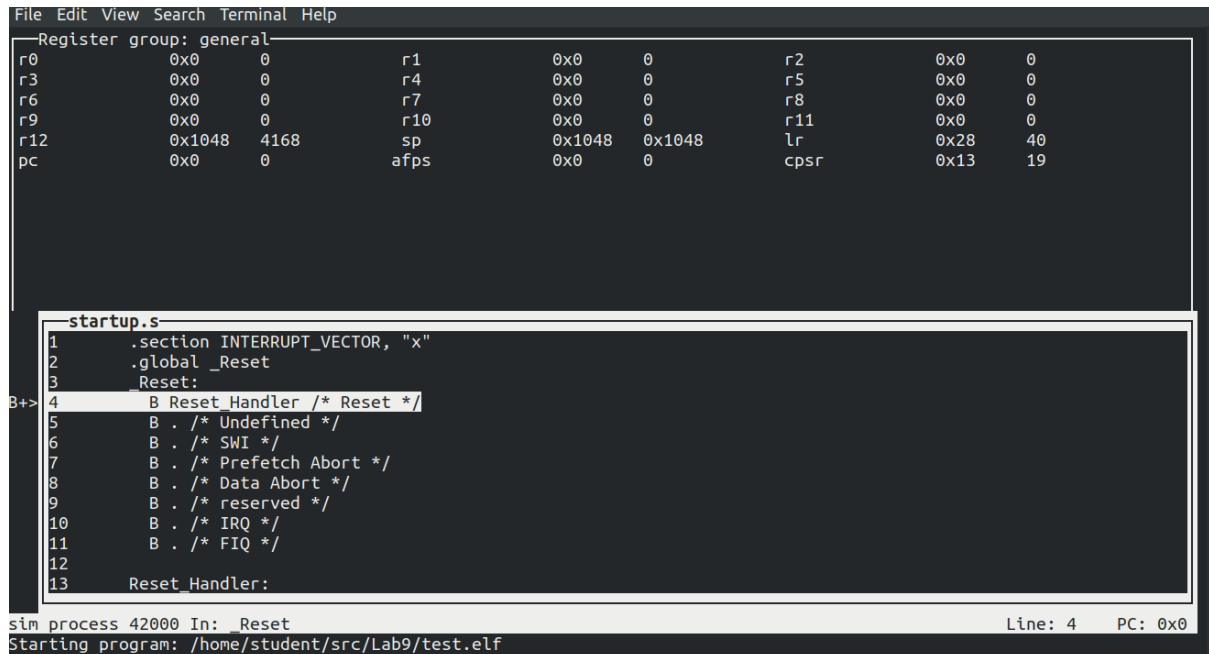
Sumário

9.2.1 Rode o Simplest Bare Metal Program	3
9.2.2 Imprima "Hello World" na placa versatile emula pelo qemu	5
9.2.3 tratando a instrução inválida em startup.s	6
9.2.4 Um Undefined Handler simples, porém errado.	8
9.2.5 A pilha no Undefined mode.	10
9.2.6 Undefined handler	11
9.2.7 modo kernel x modo usuário	12
Embora existam diversos modos no ARM, podemos classificar em usuário e o resto. Quando o ARM começa a executar está em modo supervisor. Passe para o modo usuário usando MSR e tente voltar do modo usuário para supervisor também usando MSR.	12
O que acontece? Por quê? Refaça a experiência trocando entre modos dentro do resto (undefined, abort, supervisor, etc.). O que acontece? É possível concluir então que existem dois grandes modos: usuário e supervisor?	12
Apêndice	13
9.2.1	13
startup.s	13
entrypoint.c	13
test.ld	13
9.2.2	14
test.c	14
startup.s	14
test.ld	14
9.2.3	15
startup.s	15
test.c	15
vector_table.ld	16
9.2.4	16
startup.s	16
test.c	17
vector_table.ld	17
9.2.5	18
vector_table.ld	18
9.2.6	18
vector_table.ld	18
9.2.7	18
vector_table.ld	18

9.2.1 Rode o Simplest Bare Metal Program

Executando os comandos do tutorial, obtivemos arquivos .o a partir de startup.s e de entryptoint.c. Fizemos o link com test.ld e iniciamos o gdb.

Abaixo, imagens do processo de debugging.



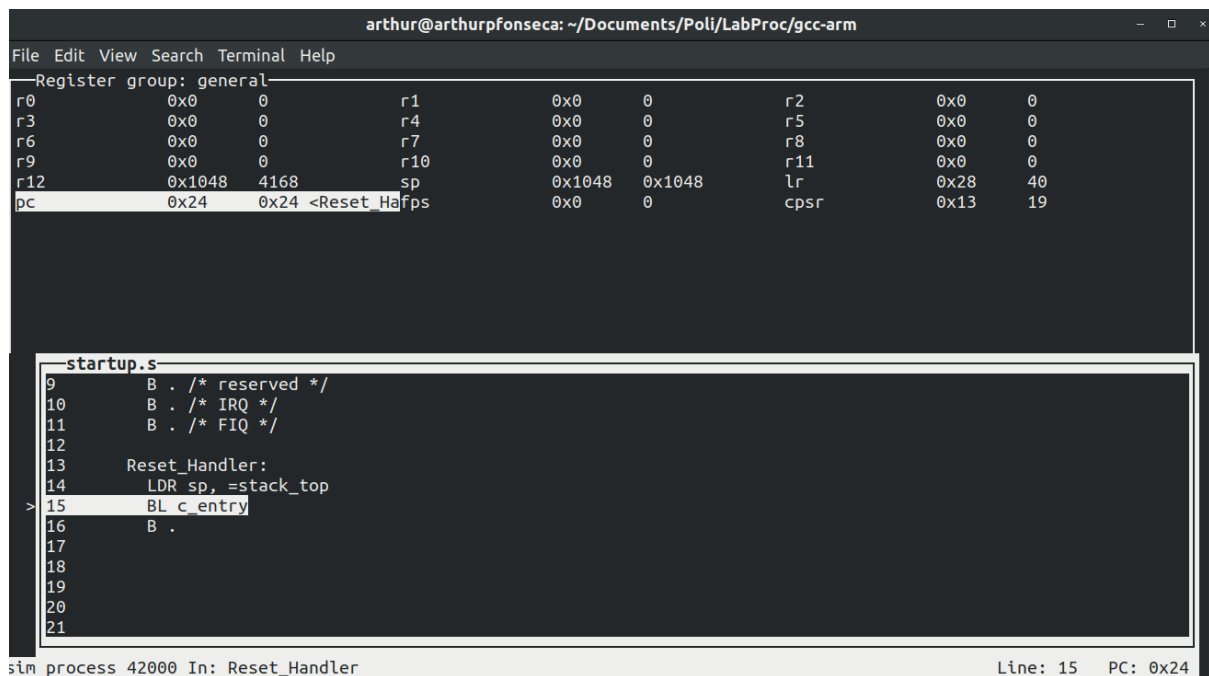
The screenshot shows the GDB interface with the 'Register group: general' window displaying the initial values of registers r0 through r12, sp, afps, r2, r5, r8, r11, lr, and cpsr. The 'startup.s' file is open, showing the assembly code for the Reset_Handler. The status bar at the bottom indicates 'sim process 42000 In: Reset' and 'Starting program: /home/student/src/Lab9/test.elf'.

```
File Edit View Search Terminal Help
Register group: general
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     0x1048   4168   sp      0x1048   0x1048   lr      0x28      40
pc      0x0      0      afps    0x0      0      cpsr    0x13      19

startup.s
1      .section INTERRUPT_VECTOR, "x"
2      .global _Reset
3      _Reset:
4      B Reset_Handler /* Reset */
5      B . /* Undefined */
6      B . /* SWI */
7      B . /* Prefetch Abort */
8      B . /* Data Abort */
9      B . /* reserved */
10     B . /* IRQ */
11     B . /* FIQ */
12
13     Reset_Handler:

sim process 42000 In: Reset Line: 4 PC: 0x0
Starting program: /home/student/src/Lab9/test.elf
```

Início da startup.s



The screenshot shows the GDB interface with the 'Register group: general' window displaying the values of registers after the Reset_Handler. The 'startup.s' file is open, showing the assembly code for the Reset_Handler. The status bar at the bottom indicates 'sim process 42000 In: Reset_Handler' and 'Line: 15 PC: 0x24'.

```
File Edit View Search Terminal Help
Register group: general
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     0x1048   4168   sp      0x1048   0x1048   lr      0x28      40
pc      0x24     0x24   <Reset_Handler> afps    0x0      0      cpsr    0x13      19

startup.s
9      B . /* reserved */
10     B . /* IRQ */
11     B . /* FIQ */
12
13     Reset_Handler:
14     LDR sp, =stack_top
15     BL c_entry
16     B .
17
18
19
20
21

sim process 42000 In: Reset_Handler Line: 15 PC: 0x24
```

Logo antes de entrar em entryptoint.s

```

arthur@arthurpfonseca: ~/Documents/Poli/LabProc/gcc-arm
File Edit View Search Terminal Help
Register group: general
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     0x1044 4164
r12     0x1048 4168 sp      0x1038 0x1038 lr      0x28 40
pc      0x44 0x44 <c_entry+fps 0x0 0 cpsr 0x13 19

entrypoint.s
5      .type    c_entry, %function
6      c_entry:
7          @ args = 0, pretend = 0, frame = 0
8          @ frame_needed = 1, uses_anonymous_args = 0
9          mov     ip, sp
10         stmfd   sp!, {fp, ip, lr, pc}
11         sub     fp, ip, #4
12         mov     r3, #0
13         mov     r0, r3
> 14         ldmfdd sp, {fp, sp, pc}
15         .size    c_entry, .-c_entry
16         .ident   "GCC: (GNU) 3.4.3"
17

sim process 42000 In: c_entry                               Line: 14   PC: 0x44
Reset Handler () at startup.s:15
```

Executando interior da entrypoint.s

9.2.2 Imprima "Hello World" na placa versatile emula pelo qemu

Desta vez, nossa função em C printa "Hello World".

```

test.c
8      }
9
10     void c_entry() {
11         print_uart0("Hello world!\n");
> 12     }
13
14
15
16
17

remote Thread 1 In: c_entry
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
c_entry () at test.c:12
(gdb) 
```

Função c_entry termina de executar.

```

ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_card_driver returned
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_concat returned error
ALSA lib confmisc.c:1251:(snd_func_refer) error evaluating name
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_refer returned error:
ALSA lib conf.c:4771:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM default
alsa: Could not initialize DAC
alsa: Failed to open 'default':
alsa: Reason: No such file or directory
audio: Failed to create voice 'lm4549.out'
Hello world!

```

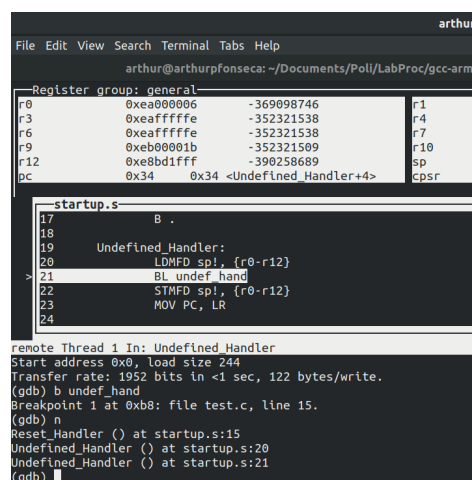
O print que sua execução gera.

9.2.3 tratando a instrução inválida em startup.s

Definimos uma instrução inválida no Reset_Handler, em startup.s (ver apêndice). Sua execução imediatamente nos leva ao Undefined_Handler, definido no mesmo arquivo, que, por sua vez, salva o estado dos registradores e, então, chama a função undef_hand, definida em test.c.

```
ALSA lib confmisc.c:1251:(snd_func_refer) error evaluating name
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_concat returned error: No such file or directory
ALSA lib conf.c:4771:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM default
alsa: Could not initialize DAC
alsa: Failed to open 'default':
alsa: Reason: No such file or directory
audio: Failed to create voice 'ln4549.out'
Hello world!
Instrucao invalida!!!!
```

Instrução Inválida sendo printado.



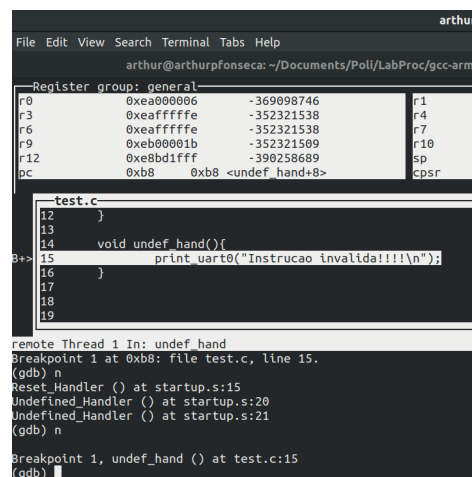
```
File Edit View Search Terminal Tabs Help
arthur@arthurfonseca: ~/Documents/Pol/LabProc/gcc-arm

Register group: general
r0 0xea000006 -369098746 r1
r3 0xeaffffff -352321538 r4
r6 0xeaffffff -352321538 r7
r9 0xeb00001b -352321509 r10
r12 0xebdbffff -390258689 sp
pc 0xb8 0xb8 <Undefined_Handler+4> cpsr

startup.s
17 B .
18
19 Undefined_Handler:
20 LDMFD sp!, {r0-r12}
21 BL undef_hand
22 STMFD sp!, {r0-r12}
23 MOV PC, LR
24

remote Thread 1 In: Undefined_Handler
Start address 0x0, load size 244
Transfer rate: 1952 bits in <1 sec, 122 bytes/write.
(gdb) b undef_hand
Breakpoint 1 at 0xb8: file test.c, line 15.
(gdb) n
Reset_Handler () at startup.s:15
Undefined_Handler () at startup.s:20
Undefined_Handler () at startup.s:21
(gdb)
```

Undefined Handler invocando handler definido em test.c



```
File Edit View Search Terminal Tabs Help
arthur@arthurfonseca: ~/Documents/Pol/LabProc/gcc-arm

Register group: general
r0 0xea000006 -369098746 r1
r3 0xeaffffff -352321538 r4
r6 0xeaffffff -352321538 r7
r9 0xeb00001b -352321509 r10
r12 0xebdbffff -390258689 sp
pc 0xb8 0xb8 <undef_hand+8> cpsr

test.c
12 }
13
14 void undef_hand(){
15     print_uart0("Instrucao invalida!!!!\\n");
16 }
17
18
19

remote Thread 1 In: undef hand
Breakpoint 1 at 0xb8: file test.c, line 15.
(gdb) n
Reset_Handler () at startup.s:15
Undefined_Handler () at startup.s:20
Undefined_Handler () at startup.s:21
(gdb) n
Breakpoint 1, undef_hand () at test.c:15
(gdb)
```

Handler em C executando após ser chamado.

```

File Edit View Search Terminal Tabs Help
arthur@arthurfonseca: ~/Documents/Pol...

Register group: general
r0      0xdc  220
r3      0x0   0
r6      0xea0ffffe  -352321538
r9      0xeb00001b  -352321509
r12     0xe8bd1fff  -390258689
pc      0x3c  0x3c  <Undefined_Handler+12>

-startup.s
19      Undefined_Handler:
20          LDWFD sp!, {r0-r12}
21          BL undef_hand
22          STMFD sp!, {r0-r12}
> 23          MOV PC, LR
24
25
26

remote Thread 1 In: Undefined_Handler
(gdb) n

Breakpoint 1, undef_hand () at test.c:15
(gdb) n
Undefined_Handler () at startup.s:22
Undefined_Handler () at startup.s:23
(gdb) s
Remote connection closed
(gdb)

```

Handler termina sua execução.

```

arthur@arthurfonseca: ~/Documents/Pol/LabProc/gcc-arm

Hello world!
Instrucao Invalida!!!!
gemu: fatal: Trying to execute code outside RAM or ROM at 0xea000008

R00=000000dc R01=ea000009 R02=0000000a R03=00000000
R04=ea0ffffe R05=ea0ffffe R06=ea0ffffe R07=ea0ffffe
R08=e59fd018 R09=eb00001b R10=fffffffb R11=000000dc
R12=e8bd1fff R13=00000008 R14=000000c0 R15=ea000008
PSR=600001fb -ZC- T und32
s00=00000000 s01=00000000 d00=0000000000000000
s02=00000000 s03=00000000 d01=0000000000000000
s04=00000000 s05=00000000 d02=0000000000000000
s06=00000000 s07=00000000 d03=0000000000000000
s08=00000000 s09=00000000 d04=0000000000000000
s10=00000000 s11=00000000 d05=0000000000000000
s12=00000000 s13=00000000 d06=0000000000000000
s14=00000000 s15=00000000 d07=0000000000000000
s16=00000000 s17=00000000 d08=0000000000000000
s18=00000000 s19=00000000 d09=0000000000000000
s20=00000000 s21=00000000 d10=0000000000000000
s22=00000000 s23=00000000 d11=0000000000000000
s24=00000000 s25=00000000 d12=0000000000000000
s26=00000000 s27=00000000 d13=0000000000000000
s28=00000000 s29=00000000 d14=0000000000000000
s30=00000000 s31=00000000 d15=0000000000000000
FPSR: 00000000
Aborted (core dumped)
student:~/src/Lab9/ex3s

```

Obervamos o valor do CPSR ao fim da execução do programa.

9.2.4 Um Undefined Handler simples, porém errado.

Sem os stack POP e PUSH, os handlers entram em loop infinito (ver último print abaixo).

```

arthur@arthurfonseca: ~/Documents/Pol/LabProc/gcc-arm

Register group: general
r0      0xd4  212
r3      0x0   0
r6      0x0   0
r9      0x0   0
r12     0x0   0
pc      0x38  0x38  <Undefined_Handler+8>
r1      0x0   0
r4      0x0   0
r7      0x0   0
r10     0x0   0
sp      0x10ec 0x10ec  1610613211
cpsr    0x600001db  1610613211
r2      0xa   10
r5      0x0   0
r8      0x0   0
r11     0x0   0
lr      0xb8  184

-startup.s
12
13      Reset_Handler:
> 14          LDR sp, =stack_top
15          BL c_entry
16          .word 0xffffffff
17          B .
18

remote Thread 1 In: Undefined_Handler
Continuing.

Breakpoint 2, Undefined_Handler () at startup.s:21
(gdb) n

Breakpoint 1, undef_hand () at test.c:15
Undefined_Handler () at startup.s:14
(gdb)

```

Início do Reset_Handler.


```
Activities Terminal jul 8 17:33
bruno@bruno-340XAA-350XAA-550XAA: ~/Documents/Poli/gcc-arm

Register group: general
r0 0x400001d3 1073742291 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 0x10fc 4348
r12 0x0 0 sp 0x10f8 0x10f8 lr 0x38 56
pc 0xa8 0xa8 <c_entry+8> cpsr 0x400001d3 1073742291

test.c
7
8
9
10 void c_entry() {
11     print_uart0("Hello world!\n");
12 }
13
14 void undefined(){
15     print_uart0("Função inválida!\n");
16     while(1);
17 }

remote Thread 1 In: c_entry Line: 11 PC: 0xa8
(gdb) s
Reset_Handler () at startup.s:15
(gdb) s
Reset_Handler () at startup.s:17
(gdb) s
Reset_Handler () at startup.s:18
(gdb) s
Reset_Handler () at startup.s:19
(gdb) s
c_entry () at test.c:11
(gdb) 
```

No ponto do print abaixo, o CPSR foi modificado para o modo Handler, porém o valor do CPSR do modo privilegiado está salvo em r0:

```
Activities Terminal jul 8 17:34
bruno@bruno-340XAA-350XAA-550XAA: ~/Documents/Poli/gcc-arm

Register group: general
r0 0xd4 212 r1 0x0 0 r2 0xa 10
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 0x20fc 8444
r12 0x0 0 sp 0x20f8 0x20f8 lr 0x3c 60
pc 0xc0 0xc0 <undefined+8> cpsr 0x600001db 1610613211

test.c
10 void c_entry() {
11     print_uart0("Hello world!\n");
12 }
13
14 void undefined(){
15     print_uart0("Função inválida!\n");
16     while(1);
17 }
18
19

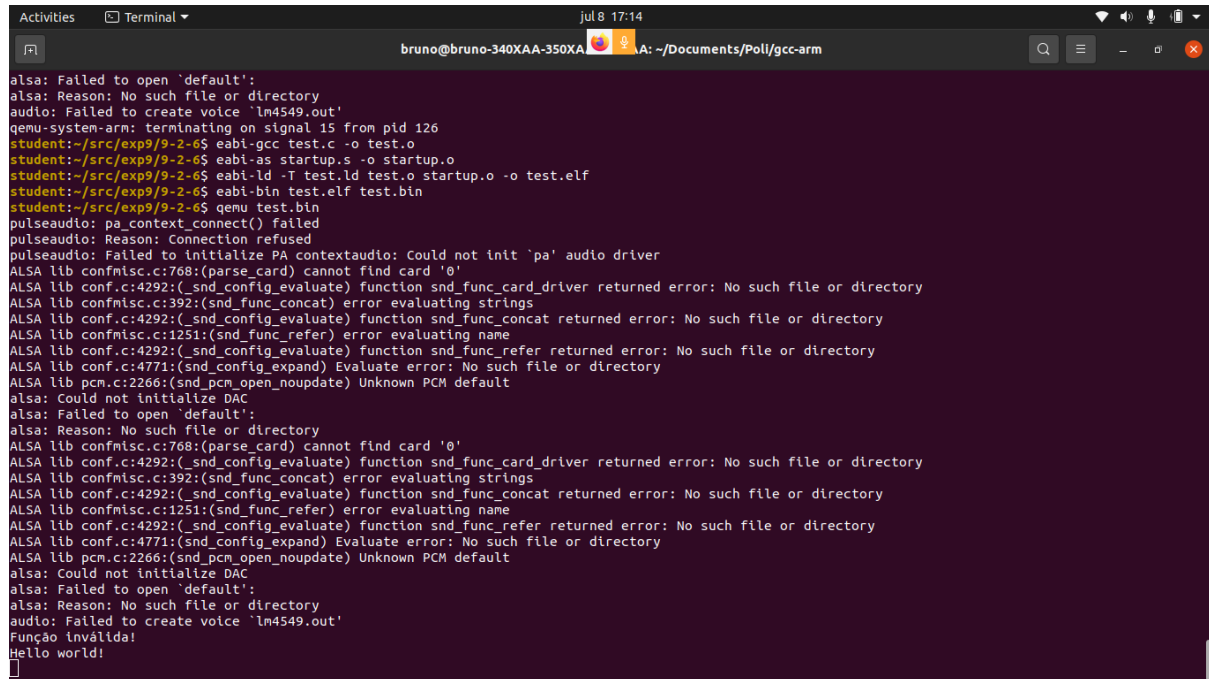
remote Thread 1 In: undefined Line: 15 PC: 0xc0
(gdb) s
(gdb) s
(gdb) s
(gdb) s
(gdb) s
c_entry () at test.c:12
(gdb) n
Undefined_Handler () at startup.s:24
(gdb) n
undefined () at test.c:14
(gdb) n
(gdb) 
```

Como respondido pelo próprio professor, a pilha de Undefined é inicializada em Reset_Handler, pois a pilha de undefined não deve ser alterada logo na entrada de Undefined_Handler - O erro é semelhante a alterar o topo da pilha na chamada de uma função.

9.2.6 Undefined handler

Quando há uma mudança de modo, o SPSR recebe o valor do CPSR anterior e com a instrução **LDMFD sp!,{R0-R12,pc}^**, o CPSR recebe o valor de SPSR, se tiver os privilégios necessários (por causa do chapeuzinho), voltando para o modo anterior, além de recuperar os registradores. Essa instrução não salva os registradores r13 e r14..

Como pode ver pelo print abaixo, o “Hello world!” foi printado depois de “Função inválida!”, assim, sabemos que voltou para o modo supervisor após o modo undefined.



```
Activities Terminal jul 8 17:14
bruno@bruno-340XAA-350XA: ~/Documents/Poll/gcc-arm

alsa: Failed to open 'default':
alsa: Reason: No such file or directory
audio: Failed to create voice 'lm4549.out'
qemu-system-arm: terminating on signal 15 from pid 126
student:~/src/exp9/p-2-4$ eabi-gcc test.c -o test.o
student:~/src/exp9/p-2-4$ eabi-as startup.s -o startup.o
student:~/src/exp9/p-2-4$ eabi-ld -T test.ld test.o startup.o -o test.elf
student:~/src/exp9/p-2-4$ eabi-bin test.elf test.bin
student:~/src/exp9/p-2-4$ qemu test.bin
pulseaudio: pa_context_connect() failed
pulseaudio: Reason: connection refused
pulseaudio: Failed to initialize PA contextaudio: Could not init 'pa' audio driver
ALSA lib confmisc.c:768:(parse_card) cannot find card '0'
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_card_driver returned error: No such file or directory
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_concat returned error: No such file or directory
ALSA lib confmisc.c:1251:(snd_func_refer) error evaluating name
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_refer returned error: No such file or directory
ALSA lib conf.c:4771:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM default
alsa: Could not initialize DAC
alsa: Failed to open 'default':
alsa: Reason: No such file or directory
ALSA lib confmisc.c:768:(parse_card) cannot find card '0'
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_card_driver returned error: No such file or directory
ALSA lib confmisc.c:392:(snd_func_concat) error evaluating strings
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_concat returned error: No such file or directory
ALSA lib confmisc.c:1251:(snd_func_refer) error evaluating name
ALSA lib conf.c:4292:(snd_config_evaluate) function snd_func_refer returned error: No such file or directory
ALSA lib conf.c:4771:(snd_config_expand) Evaluate error: No such file or directory
ALSA lib pcm.c:2266:(snd_pcm_open_noupdate) Unknown PCM default
alsa: Could not initialize DAC
alsa: Failed to open 'default':
alsa: Reason: No such file or directory
audio: Failed to create voice 'lm4549.out'
Função inválida!
Hello world!
```

Responda, a respeito de

LDMFD sp!,{R0-R12,pc}^

Por que tem um chapeuzinho no final da instrução? Para que serve isso?

Há um chapeuzinho ao final da instrução para que o processador retorne ao contexto em que estava antes da instrução em questão.

Por que essa instrução não salva os registradores sp (ou r13) e r14?

Porque esses registradores são especiais (stack pointer e link register), e não de propósito geral como os 12 primeiros. Os registradores especiais não fazem parte do contexto no sentido de registradores que são necessários para a execução da função.

Se essa é a primeira instrução a ser executada, o sp já deve ter sido inicializado.

Quem fez isso? (você já deve ter feito isso logo quando a placa é inicializada usando a instrução MSR para chavear o modo e inicializar o sp).

Quem inicializa o sp é o Reset_Handler, com a instrução LDR sp, =stack_top.

9.2.7 modo kernel x modo usuário

Embora existam diversos modos no ARM, podemos classificar em usuário e o resto. Quando o ARM começa a executar está em modo supervisor. Passe para o modo usuário usando MSR e tente voltar do modo usuário para supervisor também usando MSR.

O que acontece? Por quê? Refaça a experiência trocando entre modos dentro do resto (undefined, abort, supervisor, etc.). O que acontece? É possível concluir então que existem dois grandes modos: usuário e supervisor?

Quando passamos para o modo usuário, não é possível voltar para o modo supervisor, pois o usuário não tem permissão para isso.

Por outro lado, a mudança entre os outros modos é possível ser feita, já que têm permissão para isso.

Assim, podemos concluir que existem dois grandes modos: usuário e supervisor.

Apêndice

9.2.1

startup.s

```
.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
    B Reset_Handler /* Reset */
    B . /* Undefined */
    B . /* SWI */
    B . /* Prefetch Abort */
    B . /* Data Abort */
    B . /* reserved */
    B . /* IRQ */
    B . /* FIQ */

Reset_Handler:
    LDR sp, =stack_top
    BL c_entry
    B .
```

entrypoint.c

```
int c_entry() {
    return 0;
}
```

test.ld

```
ENTRY(_Reset)
SECTIONS
{
    . = 0x0;
    .text : {
        startup.o (INTERRUPT_VECTOR)
        *(.text)
    }
    .data : { *(.data) }
    .bss : { *(.bss COMMON) }
    . = ALIGN(8);
    . = . + 0x1000; /* 4kB of stack memory */
    stack_top = .;
}
```

9.2.2

test.c

```
volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s) {
    while(*s != '\0') { /* Loop until end of string */
        *UART0DR = (unsigned int)(*s); /* Transmit char */
        s++; /* Next char */
    }
}

void c_entry() {
    print_uart0("Hello world!\n");
}
```

startup.s

```
.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
    B Reset_Handler /* Reset */
    B . /* Undefined */
    B . /* SWI */
    B . /* Prefetch Abort */
    B . /* Data Abort */
    B . /* reserved */
    B . /* IRQ */
    B . /* FIQ */

Reset_Handler:
    LDR sp, =stack_top
    BL c_entry
    B .
```

test.ld

```
ENTRY(_Reset)
SECTIONS
{
    . = 0x0;
    .text : {
        startup.o (INTERRUPT_VECTOR)
        *(.text)
    }
    .data : { *(.data) }
```

```

.bss : { *(.bss COMMON) }
. = ALIGN(8);
. = . + 0x1000; /* 4kB of stack memory */
stack_top = .;
}

```

9.2.3

startup.s

```

.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
    B Reset_Handler /* Reset */
    B Undefined_Handler /* Undefined */
    B . /* SWI */
    B . /* Prefetch Abort */
    B . /* Data Abort */
    B . /* reserved */
    B . /* IRQ */
    B . /* FIQ */

Reset_Handler:
    LDR sp, =stack_top
    BL c_entry
    .word 0xffffffff
    B .

Undefined_Handler:
    LDMFD sp!, {r0-r12}
    BL undef_hand
    STMFD sp!, {r0-r12}
    MOV PC, LR

```

test.c

```

volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s) {
    while(*s != '\0') { /* Loop until end of string */
        *UART0DR = (unsigned int)(*s); /* Transmit char */
        s++; /* Next char */
    }
}

```

```

void c_entry() {
    print_uart0("Hello world!\n");
}

void undef_hand(){
    print_uart0("Instrucao invalida!!!!\n");
}

```

vector_table.ld

```

ENTRY(_Reset)
SECTIONS
{
    . = 0x0;
    .text : {
        startup.o (INTERRUPT_VECTOR)
        *(.text)
    }
    .data : { *(.data) }
    .bss : { *(.bss) }
    . = . + 0x1000; /* 4kB of stack memory */
    stack_top = .;
}

```

9.2.4

startup.s

```

.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
    B Reset_Handler /* Reset */
    B Undefined_Handler /* Undefined */
    B . /* SWI */
    B . /* Prefetch Abort */
    B . /* Data Abort */
    B . /* reserved */
    B . /* IRQ */
    B . /* FIQ */

Reset_Handler:
    LDR sp, =stack_top
    BL c_entry
    .word 0xffffffff
    B .

Undefined_Handler:

```

```
LDR sp, =stack_top
BL undef_hand
```

test.c

```
volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s) {
    while(*s != '\0') { /* Loop until end of string */
        *UART0DR = (unsigned int)(*s); /* Transmit char */
        s++; /* Next char */
    }
}

void c_entry() {
    print_uart0("Hello world!\n");
}

void undef_hand(){
    print_uart0("Instrucao invalida!!!!\n");
}
```

vector_table.ld

```
ENTRY(_Reset)
SECTIONS
{
    . = 0x0;
    .text : {
        startup.o (INTERRUPT_VECTOR)
        *(.text)
    }
    .data : { *(.data) }
    .bss : { *(.bss) }
    . = . + 0x1000; /* 4kB of stack memory */
    stack_top = .;
}
```

9.2.5

vector_table.ld

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

```

.text : {
startup.o (INTERRUPT_VECTOR)
*(.text)
}
.data : { *(.data) }
.bss : { *(.bss) }
. = . + 0x1000; /* 4kB of stack memory */
stack_top = .;
}

```

test.c

```

volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s) {
while(*s != '\0') { /* Loop until end of string */
*UART0DR = (unsigned int)(*s); /* Transmit char */
s++; /* Next char */
}
}

void c_entry() {
print_uart0("Hello world!\n");
}

void undefined(){
print_uart0("Função inválida!\n");
while(1);
}

```

startup.s

```

.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
B Reset_Handler /* Reset */
B Undefined_Handler /* Undefined */
B . /* SWI */
B . /* Prefetch Abort */
B . /* Data Abort */
B . /* reserved */
B . /* IRQ */
B . /* FIQ */

Reset_Handler:
LDR sp, =stack_top
MRS r0, cpsr

```



```

MSR cpsr_ctl, #0b11011011
LDR sp, =undefined_stack_top
MSR cpsr, r0
BL c_entry
.word 0xffffffff
B .

```

```

Undefined_Handler:
B undefined

```

9.2.6

startup.s

```

.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
    B Reset_Handler /* Reset */
    B Undefined_Handler /* Undefined */
    B . /* SWI */
    B . /* Prefetch Abort */
    B . /* Data Abort */
    B . /* reserved */
    B . /* IRQ */
    B . /* FIQ */

Reset_Handler:
    LDR sp, =stack_top

    MRS r0, cpsr           @ salvando o modo corrente em R0
    MSR cpsr_ctl, #0b11011011 @ alterando o modo para undefined - o SP eh
    automaticamente chaveado ao chavear o modo
    LDR sp, =undefined_stack_top @ a pilha de undefined eh setada
    MSR cpsr, r0           @ volta para o modo anterior

    LDMFD sp!, {lr}

    .word 0xffffffff
    BL c_entry

    B .

Undefined_Handler:
    STMFD sp!, {R0-R12, lr}
    BL undef_hand
    LDMFD sp!, {R0-R12, pc}^

```

```
@MOV PC, LR
```

test.c

```
volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s) {
    while(*s != '\0') { /* Loop until end of string */
        *UART0DR = (unsigned int)(*s); /* Transmit char */
        s++; /* Next char */
    }
}

void c_entry() {
    print_uart0("Hello world!\n");
}

void undef_hand(){
    print_uart0("Instrucao invalida!!!!\n");
}
```

vector_table.ld

```
ENTRY(_Reset)
SECTIONS
{
    . = 0x0;
    .text : {
        startup.o (INTERRUPT_VECTOR)
        *(.text)
    }
    .data : { *(.data) }
    .bss : { *(.bss) }
    . = . + 0x1000; /* 4kB of stack memory */
    stack_top = .;
}
```

9.2.7

vector_table.ld

```
ENTRY(_Reset)
SECTIONS
{
```

```

. = 0x0;
.text : {
startup.o (INTERRUPT_VECTOR)
*(.text)
}
.data : { *(.data) }
.bss : { *(.bss) }
. = . + 0x1000; /* 4kB of stack memory */
stack_top = .;
}

```

startup.s

```

.section INTERRUPT_VECTOR, "x"
.global _Reset
_Reset:
    B Reset_Handler /* Reset */
    B Undefined_Handler /* Undefined */
    B . /* SWI */
    B . /* Prefetch Abort */
    B . /* Data Abort */
    B . /* reserved */
    B . /* IRQ */
    B . /* FIQ */

Reset_Handler:
    LDR sp, =stack_top
    MRS r0, cpsr
    MSR cpsr_ctl, #0b11011011
    LDR sp, =undefined_stack_top
    MSR cpsr, r0
    .word 0xffffffff
    BL c_entry
    B .

Undefined_Handler:
    STMFD sp!, {r0-r12, lr}
    B undefined
    LDMFD sp!, {r0-r12, pc}^

```

test.c

```

volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s) {
    while(*s != '\0') { /* Loop until end of string */
        *UART0DR = (unsigned int)(*s); /* Transmit char */
    }
}

```

```
s++; /* Next char */  
}  
  
void c_entry() {  
    print_uart0("Hello world!\n");  
}  
  
void undefined(){  
    print_uart0("Função inválida!\n");  
}
```