Systèmes Embarqués 1 & 2

Classes T-2/I-2 // 2018-2019

p.05 – Architecture interne

Solutions

Exercice 1

Trouver et décrire plusieurs techniques de désassemblage de code assembleur

Solution:

- Il existe plusieurs techniques pour désassembler du code
  - Taper le code binaire dans la mémoire de la cible et ensuite à l'aide du débugger désassembler le code
  - Entrer le code binaire dans un fichier assembleur, compiler le programme et à l'aide de l'utilitaire «objdump» désassembler le code
  - Identique à 2. mais dans un programme C

Exercice 2

Désassembler les instructions ci-dessous

```
<address>:   <instruction>
 80001000:      e1a00000
 80001004:      e3a0000a
 80001008:      e59f1018
 8000100c:      e1d110b0
 80001010:      e59f3014
 80001014:      e5932000

 80001018:      e0822001
 8000101c:      e2500001
 80001020:      1afffffc

 80001024:      e12fff1e

 80001028:      80002068
 8000102c:      80002064
```

Solution:

- taper le code ci-dessous dans un fichier t.S

```
        .text
.long   0xe1a00000
.long   0xe3a0000a
.long   0xe59f1018
.long   0xe1d110b0
.long   0xe59f3014
.long   0xe5932000

.long   0xe0822001
.long   0xe2500001
.long   0x1afffffc

.long   0xe12fff1e

.long   0x80002068
.long   0x80002064
```

- assembler le code avec la commande `arm-none-eabi-gcc -c -o t.o t.S`
- désassembler le code binaire avec la commande `arm-none-eabi-objdump -Ds t.o`

```
 0:   e1a00000   nop                     ; (mov r0, r0)
 4:   e3a0000a   mov    r0, #10
 8:   e59f1018   ldr    r1, [pc, #24]   ; 0x28
 c:   e1d110b0   ldrh   r1, [r1]
10:   e59f3014   ldr    r3, [pc, #20]   ; 0x2c
14:   e5932000   ldr    r2, [r3]
18:   e0822001   add    r2, r2, r1
1c:   e2500001   subs   r0, r0, #1
20:   1afffffc   bne    0x18
24:   e12fff1e   bx     lr
28:   80002068                          ; addresse d'une variable
2c:   80002064                          ; addresse d'une variable
```

- décodage de l'instruction