3 - La pile d'exécution

Nous allons nous intéresser à ce qui se passe sur la pile d'exécution lors d'un appel de fonction.
Le code source et le code compilé qui en résulte sont au verso.

1. Placez-vous dans un répertoire **`M2101/FunctionCall`**, récupérez le fichier **`function_call.c`** sur le site Web du cours, et complétez le tableau suivant à l'aide de **`gdb`** :

arrêt	*variable*	*adresse*	*valeur*
l. 14	**`main`**		
	`to_seconds`		
	`rip`	-	
	`$rip`		
	`rsp`	-	
	`rbp`	-	
	`hh`		
	`mm`		
	`ss`		
	`total`		
l. 4	**`edi`**	-	
	`esi`	-	
	`edx`	-	
	`rip`	-	
	`rsp`	-	
	`rbp`	-	
	`hours`		
	`minutes`		
	`seconds`		
	`result`		
	-	**`backtrace`**	
Faites un dessin de l'état de la pile (prenez une page A4 complète), avec les cadres des fonctions, en y mettant les noms, les adresses et les valeurs des variables (un **`x/10gx $rsp`** peut vous aider).			
l. 15	**`total`**		
	`eax`	-	
	`rip`	-	
	`rsp`	-	
	`rbp`	-	
	`hh`		
	-	**`backtrace`**	

2. La fonction **`to_seconds()`** travaille-t-elle directement sur **`hh`**, **`mm`**, **`ss`** et **`total`** ?
3. Comment les paramètres sont-ils passés à la fonction **`to_seconds()`** ?
4. Comment la valeur retournée est-elle transmise à **`main()`** ?
5. Relancez le débogage et déterminez l'adresse de retour de **`to_seconds()`** et de **`main()`**.
6. Où ces adresses de retours sont-elles stockées ? Faites-les apparaître sur votre dessin.
7. À quoi correspond l'adresse qui est empilée juste au-dessus de l'adresse de retour ?

Listing 1 – function_call.c

```c
#include <stdio.h>

int to_seconds (int hours, int minutes, int seconds) {
    int result = 3600 * hours + 60 * minutes + seconds;
    hours = minutes = seconds = 0;
    printf ("Cela fait %d secondes.\n", result);
    return result;
}

int main(void) {
    int hh = 10;
    int mm = 33;
    int ss = 28;
    int total = to_seconds (hh, mm, ss);
    return 0;
}
```

Listing 2 – main() compilée

```
0000000000000692 <main>:
 692:   55                      push   rbp
 693:   48 89 e5                mov    rbp,rsp
 696:   48 83 ec 10             sub    rsp,0x10
 69a:   c7 45 f0 0a 00 00 00    mov    DWORD PTR [rbp-0x10],0xa
 6a1:   c7 45 f4 21 00 00 00    mov    DWORD PTR [rbp-0xc],0x21
 6a8:   c7 45 f8 1c 00 00 00    mov    DWORD PTR [rbp-0x8],0x1c
 6af:   8b 55 f8                mov    edx,DWORD PTR [rbp-0x8]
 6b2:   8b 4d f4                mov    ecx,DWORD PTR [rbp-0xc]
 6b5:   8b 45 f0                mov    eax,DWORD PTR [rbp-0x10]
 6b8:   89 ce                   mov    esi,ecx
 6ba:   89 c7                   mov    edi,eax
 6bc:   e8 79 ff ff ff          call   63a <to_seconds>
 6c1:   89 45 fc                mov    DWORD PTR [rbp-0x4],eax
 6c4:   b8 00 00 00 00          mov    eax,0x0
 6c9:   c9                      leave
 6ca:   c3                      ret
 6cb:   0f 1f 44 00 00          nop    DWORD PTR [rax+rax*1+0x0]
```

Listing 3 – to_seconds() compilée

```
000000000000063a <to_seconds>:
 63a:   55                      push   rbp
 63b:   48 89 e5                mov    rbp,rsp
 63e:   48 83 ec 20             sub    rsp,0x20
 642:   89 7d ec                mov    DWORD PTR [rbp-0x14],edi
 645:   89 75 e8                mov    DWORD PTR [rbp-0x18],esi
 648:   89 55 e4                mov    DWORD PTR [rbp-0x1c],edx
 64b:   8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
 64e:   69 d0 10 0e 00 00       imul   edx,eax,0xe10
 654:   8b 45 e8                mov    eax,DWORD PTR [rbp-0x18]
 657:   6b c0 3c                imul   eax,eax,0x3c
 65a:   01 c2                   add    edx,eax
 65c:   8b 45 e4                mov    eax,DWORD PTR [rbp-0x1c]
 65f:   01 d0                   add    eax,edx
 661:   89 45 fc                mov    DWORD PTR [rbp-0x4],eax
 664:   c7 45 e4 00 00 00 00    mov    DWORD PTR [rbp-0x1c],0x0
 66b:   8b 45 e4                mov    eax,DWORD PTR [rbp-0x1c]
 66e:   89 45 e8                mov    DWORD PTR [rbp-0x18],eax
 671:   8b 45 e8                mov    eax,DWORD PTR [rbp-0x18]
 674:   89 45 ec                mov    DWORD PTR [rbp-0x14],eax
 677:   8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
 67a:   89 c6                   mov    esi,eax
 67c:   48 8d 3d d1 00 00 00    lea    rdi,[rip+0xd1]
 683:   b8 00 00 00 00          mov    eax,0x0
 688:   e8 93 fe ff ff          call   520 <printf@plt>
 68d:   8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
 690:   c9                      leave
 691:   c3                      ret
```