Commencé le	Monday 16 December 2024, 09:06
État	Terminé
Terminé le	Monday 16 December 2024, 09:36
Temps mis	30 min 5 s
Points	16,17/21,00
Note	**15,40** sur 20,00 (**76,98**%)

Question **1**

Correct

Note de 1,50 sur 1,50

What happens to the stack pointer (RSP) when a value is pushed onto the stack in x86-64 architecture?

- a. Decreases by the size of the pushed value ✔
- b. Moves to a predefined address
- c. Increases by the size of the pushed value
- d. Stays unchanged

Votre réponse est correcte.

La réponse correcte est :
Decreases by the size of the pushed value

Question **2**

Incorrect

Note de 0,00 sur 1,50

In **one** word, what is the difference between REPZ and REPE

Réponse : Condition ✖

La réponse correcte est : None

Question **3**

Correct

Note de 1,50 sur 1,50

Consider the following assembly code :

jge 0x4060d1

What is the name of the register on x86-64 that will be read by this instruction?

Réponse : rflags ✔

La réponse correcte est : rflags

Question **4**

Correct

Note de 1,50 sur 1,50

Which register(s) are read during a REP instruction ?

- a. %rdx
- b. %r8
- c. %rcx ✔
- d. %rflags

Votre réponse est correcte.

La réponse correcte est :
%rcx

Question **5**

Correct

Note de 1,50 sur 1,50

Pick the most likely sample.

The next four questions pertain to the following four code samples.

f1

```
f1:
       subq    $8, %rsp
       call    callfunc
       movl    %eax, %edx
       leal    1(%rax,%rax,2), %eax
       testb   $1, %dl
       jne     .L3
       movl    %edx, %eax
       shrl    $31, %eax
       addl    %edx, %eax
       sarl    %eax
.L3:
       addq    $8, %rsp
       ret
```

f2

```
f2:
       pushq   %rbx
       xorl    %ebx, %ebx
.L3:
       movl    %ebx, %edi
       addl    $1, %ebx
       call    callfunc
       cmpl    $10, %ebx
       jne     .L3
       popq    %rbx
       ret
```

f3

```
f3:
        subq    $8, %rsp
        call    callfunc
        subl    $97, %eax
        cmpb    $4, %al
        ja      .L2
        movzbl  %al, %eax
        jmp     *.L4(,%rax,8)
.L4:
        .quad   .L3
        .quad   .L9
        .quad   .L6
        .quad   .L7
        .quad   .L8
.L3:
        movl    $42, %edx
        jmp     .L5
.L6:
        movl    $4096, %edx
        jmp     .L5
.L7:
        movl    $52, %edx
        jmp     .L5
.L8:
        movl    $6440, %edx
        jmp     .L5
.L2:
        movl    $0, %edx
        jmp     .L5
.L9:
        movl    $61, %edx
.L5:
        movl    $.LC0, %esi
        movl    $1, %edi
        movl    $0, %eax
        call    __printf_chk
        addq    $8, %rsp
        ret
.LC0:
        .string "Sum = %d\n"
```

f4

```
f4:
        subq    $40, %rsp
        movl    $1, (%rsp)
        movl    $0, 16(%rsp)
.L2:
        leaq    16(%rsp), %rsi
        movq    %rsp, %rdi
        call    callfunc
        movl    16(%rsp), %eax
        cmpl    %eax, (%rsp)
        jne     .L2
        addq    $40, %rsp
        ret
```

Which sample contains a for loop?	f2 ✔
Which sample contains a switch statement?	f3 ✔
Which sample contains a while loop?	f4 ✔
Which sample contains only an if/else construct?	f1 ✔

La réponse correcte est :

Which sample contains a for loop? → f2,

Which sample contains a switch statement? → f3,

Which sample contains a while loop? → f4,

Which sample contains only an if/else construct? → f1

Question **6**

Correct

Note de 1,50 sur 1,50

A user assembly program pushes the syscall function address on to the stack prior calling the syscall instruction.

Veuillez choisir une réponse.

- Vrai
- Faux ✔

La réponse correcte est « Faux ».

Question **7**

Correct

Note de 1,00 sur 1,00

Indicate the most likely type(s) of the data being accessed: `movl -28(%rbp), %rdx`.

- a. Array of int or unsigned int
- b. char *
- c. int
- d. We can not know
- e. unsigned short
- f. unsigned char
- g. It does not compile ✔

La réponse correcte est :
It does not compile

Question **8**

Correct

Note de 1,00 sur 1,00

Indicate the most likely type(s) of the data being accessed: `movb -8(%rbp), %dl`.

- [ ] a. We can not know
- [ ] b. Array of int or unsigned int
- [x] c. unsigned char ✔
- [ ] d. unsigned short
- [ ] e. int
- [ ] f. It does not compile
- [ ] g. char *

La réponse correcte est : unsigned char

Question **9**

Correct

Note de 1,00 sur 1,00

Which register is most likely to be used for the `long` data type?

- (•) a. rdx ✔
- ( ) b. bx
- ( ) c. eax
- ( ) d. the answer is not listed
- ( ) e. cl

La réponse correcte est : rdx

Question **10**

Correct

Note de 1,00 sur 1,00

Which register is most likely to be used for the `char` data type?

- ( ) a. rdx
- ( ) b. bx
- (•) c. cl ✔
- ( ) d. the answer is not listed
- ( ) e. eax

La réponse correcte est : cl

Question **11**

Correct

Note de 0,50 sur 0,50

How many byte(s) can be stored in the register `%ebx`?

Réponse : 4 ✔

La réponse correcte est : 4

Question **12**

Correct

Note de 0,50 sur 0,50

How many byte(s) can be stored in the register `%r9d`?

Réponse : 4 ✔

La réponse correcte est : 4

Question **13**

Correct

Note de 0,50 sur 0,50

How many byte(s) can be stored in the register `%eax`?

Réponse : 4 ✔

La réponse correcte est : 4

Question **14**

Correct

Note de 0,50 sur 0,50

How many byte(s) can be stored in the register `%edx`?

Réponse : 4 ✔

La réponse correcte est : 4

Question **15**

Incorrect

Note de 0,00 sur 1,00

What is the idiomatic way to get the quotient of an unsigned integer division by two in x86 assembly ?

a. shr $1, %rax

b. movq $2, %rbx
idivq %rbx

c. sal $1, %rax

d. shl $1, %rax

e. sar $1, %rax ✖

Votre réponse est incorrecte.

La réponse correcte est :
shr $1, %rax

Question **16**

Correct

Note de 1,00 sur 1,00

After a function call, the stack is **always** unaligned.

Veuillez choisir une réponse.

Vrai

Faux ✔

La réponse correcte est « Faux ».

Question **17**

Correct

Note de 1,00 sur 1,00

Is the following code valid according to the ABI?

```
    .global main
    .text
main:
    push %rbp
    mov %rsp, %rbp
    mov str, %rdi
    call puts
    mov $0, %eax
    pop %rbp
    ret

    .section .data
str:
    .asciz "Hello!"
```

Veuillez choisir une réponse.

- Vrai
- Faux ✔

The issue is :

mov str, %rdi

It copies the string's bytes in to %rdi, but puts expect a pointer.\

Correct code is :

mov $str, %rdi

In this case the source will as well be an immediate value but with the address of the label str which is a pointer.

La réponse correcte est « Faux ».

Question **18**

Incorrect

Note de 0,00 sur 1,00

Is the following stack pointer aligned ?

```
rsp: 0x7fffffff8884
```

Veuillez choisir une réponse.

- Vrai ✖
- Faux

La réponse correcte est « Faux ».

Question **19**

Partiellement correct

Note de 0,67 sur 2,00

In the following questions, we give you C code and a portion of the assembly generated by some compiler for that code. (Sometimes we blank out a part of the assembly.) The C code contains a variable, constant, or function called `waldo`, and a point in the assembly is marked with asterisks `***`. Your job is to find Waldo: write an **assembly expression or constant** that holds the value of `waldo` at the marked point. We've done the first one for you.

NON-QUESTION: Where's Waldo?

```
int identity(int waldo) {
    return waldo;
}
```

```
00000000004007f6 <identity>:
  4007f6:       55                      push   %rbp
  4007f7:       48 89 e5                mov    %rsp,%rbp
  4007fa:       89 7d fc                mov    %edi,-0x4(%rbp)
  4007fd:       8b 45 fc                mov    -0x4(%rbp),%eax
           ***
  400800:       5d                      pop    %rbp
  400801:       c3                      retq
```

ANSWER: `%edi`, `-0x4(%rbp)`, `%eax`, and `%rax` all hold the value of `waldo` at the marked point, so any of them is a valid answer. If the asterisks came before the *first* instruction, only `%edi` would work.

```
int int_array_get(int* a, int waldo) {
    int x = a[waldo];
    return x;
}
```

```
00000000004007d9 <int_array_get>:
INSTRUCTIONS OMITTED
          ***
 4007dc:       8b 04 b7                mov    (%rdi,%rsi,4),%eax
 4007df:       c3                      retq
```

(%rdi,%rsi,8) ✖

```
int matrix_get(int** matrix, int row, int col) {
    int* waldo = matrix[row];
    return waldo[col];
}
```

```
00000000004007e0 <matrix_get>:
 4007e0:       48 63 f6                movslq %esi,%rsi
 4007e3:       48 63 d2                movslq %edx,%rdx
          ***
 4007e6:       ?? ?? ?? ??             mov    ??,%rax
 4007ea:       8b 04 90                mov    (%rax,%rdx,4),%eax
 4007ed:       c3                      retq
```

%rsi ✖

```c
int f1(int a, int b, int waldo, int d) {
    if (a > b) {
        return waldo;
    } else {
        return d;
    }
}
```

```
0000000000400802 <f1>:
           ***
  400802:       55                      push   %rbp
  400803:       48 89 e5                mov    %rsp,%rbp
  400806:       89 7d fc                mov    %edi,-0x4(%rbp)
  400809:       89 75 f8                mov    %esi,-0x8(%rbp)
  40080c:       89 55 f4                mov    %edx,-0xc(%rbp)
  40080f:       89 4d f0                mov    %ecx,-0x10(%rbp)
  400812:       8b 45 fc                mov    -0x4(%rbp),%eax
  400815:       3b 45 f8                cmp    -0x8(%rbp),%eax
  400818:       7e 05                   jle    40081f <f1+0x1d>
  40081a:       8b 45 f4                mov    -0xc(%rbp),%eax
  40081d:       eb 03                   jmp    400822 <f1+0x20>
  40081f:       8b 45 f0                mov    -0x10(%rbp),%eax
  400822:       5d                      pop    %rbp
  400823:       c3                      retq
```

%edx ✔

La réponse correcte est :

```c
int int_array_get(int* a, int waldo) {
    int x = a[waldo];
    return x;
}
```

```
00000000004007d9 <int_array_get>:
INSTRUCTIONS OMITTED
           ***
 4007dc:       8b 04 b7                mov    (%rdi,%rsi,4),%eax
 4007df:       c3                      retq
```

→ %rsi,

```c
int matrix_get(int** matrix, int row, int col) {
    int* waldo = matrix[row];
    return waldo[col];
}
```

```
00000000004007e0 <matrix_get>:
 4007e0:       48 63 f6                movslq %esi,%rsi
 4007e3:       48 63 d2                movslq %edx,%rdx
          ***
 4007e6:       ?? ?? ?? ??             mov    ??,%rax
 4007ea:       8b 04 90                mov    (%rax,%rdx,4),%eax
 4007ed:       c3                      retq
```

→ (%rdi,%rsi,8),

```c
int f1(int a, int b, int waldo, int d) {
    if (a > b) {
        return waldo;
    } else {
        return d;
    }
}
```

```
0000000000400802 <f1>:
          ***
  400802:       55                      push   %rbp
  400803:       48 89 e5                mov    %rsp,%rbp
  400806:       89 7d fc                mov    %edi,-0x4(%rbp)
  400809:       89 75 f8                mov    %esi,-0x8(%rbp)
  40080c:       89 55 f4                mov    %edx,-0xc(%rbp)
  40080f:       89 4d f0                mov    %ecx,-0x10(%rbp)
  400812:       8b 45 fc                mov    -0x4(%rbp),%eax
  400815:       3b 45 f8                cmp    -0x8(%rbp),%eax
  400818:       7e 05                   jle    40081f <f1+0x1d>
  40081a:       8b 45 f4                mov    -0xc(%rbp),%eax
  40081d:       eb 03                   jmp    400822 <f1+0x20>
  40081f:       8b 45 f0                mov    -0x10(%rbp),%eax
  400822:       5d                      pop    %rbp
  400823:       c3                      retq
```

→ %edx

◄ Linux System V Syscall Table

Aller à...