4.0 - Week 4 - Workshop (MA)

Learning Objectives

- Understanding local variables in MIPS.
- Understanding function calling conventions in MIPS
- Understanding function return conventions in MIPS

Week 4 Padlet Discussion Board link: https://monashmalaysia.padlet.org/fermi/2022week4

Stack and frame pointers

Tick all the correct answers.

Question 1 *Submitted Aug 14th 2022 at 2:14:27 pm*

The stack pointer $sp:

- ☐ **A.** Always points to the saved $ra of the main function.
- ☐ **B.** Is always equal to $fp of the current function
- ✓ **C.** Decreases when a function is called.
- ☐ **D.** Always points to the next address in the stack that can be written into (but it may contain garbage).
- ☐ **E.** Is automatically updated by the system.
- ✓ **F.** Does not need to be saved on the stack between two function calls.

Question 2 *Submitted Aug 14th 2022 at 2:18:16 pm*

The frame pointer $fp:

- ✓ **A.** Always point to the saved $fp of the previous frame, if there is one.
- ☐ **B.** Is always equal the frame number, stored as a 32 bits integer, which is equal to 0 if we are in the main function.
- ☐ **C.** Always has the same value in the same function.
- ☐ **D.** Can be used with a negative shift (with for instance -4($fp)) to access an argument of the current function.

Memory diagram and calling convention

Consider the following Python code:

```
import typing

def following(a:int, b:int) -> int:
    c = a-b
    d = a*b
    return d//c
```

Question 1 *Submitted Aug 14th 2022 at 2:20:00 pm*

Which of the memory diagrams below correspond(s) to a correct faithful translation of the *following* function? (here the values in the "middle" column do not matter - they are not represented).

☐ **A**

```
-------------------------|          | 0x7FFF0390
                       a |          | 0x7FFF0394
                       b |          | 0x7FFF0398
                       c |          | 0x7FFF039C
                       d |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

☐ **B**

```
-------------------------|          | 0x7FFF0388
                       c |          | 0x7FFF038C
                       d |          | 0x7FFF0390
                       b |          | 0x7FFF0394
                       a |          | 0x7FFF0398
               saved $fp |          | 0x7FFF039C
               saved $ra |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

[✓] **C**

```
-------------------------|          | 0x7FFF0388
                       d |          | 0x7FFF038C
                       c |          | 0x7FFF0390
               saved $fp |          | 0x7FFF0394
               saved $ra |          | 0x7FFF0398
                       b |          | 0x7FFF039C
                       a |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

[ ] **D**

```
-------------------------|          | 0x7FFF0388
                       a |          | 0x7FFF038C
                       b |          | 0x7FFF0390
               saved $fp |          | 0x7FFF0394
               saved $ra |          | 0x7FFF0398
                       c |          | 0x7FFF039C
                       d |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

[✓] **E**

```
-------------------------|          | 0x7FFF0388
                       c |          | 0x7FFF038C
                       d |          | 0x7FFF0390
               saved $fp |          | 0x7FFF0394
               saved $ra |          | 0x7FFF0398
                       a |          | 0x7FFF039C
                       b |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

[ ] **F**

```
-------------------------|          | 0x7FFF0388
                       c |          | 0x7FFF038C
                       d |          | 0x7FFF0390
               saved $ra |          | 0x7FFF0394
               saved $fp |          | 0x7FFF0398
                       b |          | 0x7FFF039C
                       a |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

☐ **G**

```
-------------------------|          | 0x7FFF0388
                       a |          | 0x7FFF038C
                       b |          | 0x7FFF0390
                saved $ra |          | 0x7FFF0394
                saved $fp |          | 0x7FFF0398
                       c |          | 0x7FFF039C
                       d |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

Question 2 *Submitted Aug 14th 2022 at 2:21:23 pm*

Within the `following` function, what is the value of $fp?

0x7FFF0394

Callees in MIPS

Consider the following Python code:

```
import typing

def following(a:int, b:int) -> int:
    c = a-b
    d = a*b
    return d//c
```

Faithfully translate the `following` function into a properly commented MIPS program using the file provided by **replacing** the TODO lines with your code. You can "mark" this to check whether your function passes the tests.

We will use the diagram of the stack frame of the `following` function as shown below:

```
-------------------------|          | 0x7FFF0388
                       d |          | 0x7FFF038C
                       c |          | 0x7FFF0390
     $fp --->  saved $fp |          | 0x7FFF0394
               saved $ra |          | 0x7FFF0398
                       b |          | 0x7FFF039C
                       a |          | 0x7FFF03A0
-------------------------|          | 0x7FFF03A4
                         |          | 0x7FFF03A8
```

(Note that the addresses on the right are just an example: they will vary depending on the state of the stack at the point where the function is called.)

For convenience, we will write this as a comment in the code in this condensed and more useful format:

> `# d is at -8($fp)`
>
> `# c is at -4($fp)`
>
> `# saved fp is at ($fp)`
>
> `# saved ra is at +4($fp)`
>
> `# b is at +8($fp)`
>
> `# a is at +12($fp)`

Note that we restrict ourselves to $a > b$ cases to avoid getting into the case where Python's `//` and MIPS's `div` disagree.

Callers in MIPS

Consider the following Python code:

```
import typing

def main() -> None:
    x = int(input("Enter integer: "))
    y = int(input("Enter integer: "))
    print(following(x, y))

#in Python there is no default "main" function
#we need to indicate what to do if this file is run.
if __name__=="__main__":
    main()
```

Faithfully translate the `main` function into a properly commented MIPS program using the file provided by **replacing** the TODO lines with your code. You can "mark" this to check whether your function passes the tests.

For convenience, here is the stack frame diagram of the function `main`:

> `# y is at -8($fp)`
>
> `# x is at -4($fp)`

and the one of the `following` function:

> `# d is at -8($fp)`
>
> `# c is at -4($fp)`
>
> `# saved fp is at ($fp)`
>
> `# saved ra is at +4($fp)`
>
> `# b is at +8($fp)`
>
> `# a is at +12($fp)`

(where `$fp` refers to the frame pointer of `following`.)

Note (again) that we restrict ourselves to $x > y$ cases to avoid getting into the case where Python's `//` and MIPS's `div` disagree.

Feedback Form

Weekly Workshop Feedback Form

Question 1

I am enrolled in:

- ○ □□ Australia
- ○ □□ Malaysia

Question 2

What needs improvement?

No response

Question 3

What worked best?

No response

Question 4

How engaged were you by the workshop?

- ○ □□□ Very engaged
- ○ □□□ Engaged
- ○ □😐□ Not impressed
- ○ 😕😴□ Lost