

COMP2611: Computer Organization

MIPS Recursion

MIPS recursion

Recursive procedures
- examples

Exercises

- ❑ Since **procedures** are like small programs themselves, they may **need to use the registers**, and they **may also call other procedures** (nested calls)
 - ❑ If we don't **save** some of the values stored in the registers, they will be wiped each time we call a new procedure
- ❑ In MIPS, we need to save the registers by ourselves
- ❑ The perfect place for this is called a **stack**
 - a memory accessible only from the top (Last In First Out, LIFO)
 - placing things on the stack is called **push**
 - removing them is called **pop**
- ❑ **push** and **pop** are simply **storing** and **loading** words to and from a specific location in the memory pointed to by **the stack pointer \$sp** which always points to top of the stack

- ❑ The Caller to a nested function call performs the same steps as to a simple function call. E.g. `jal nestedProcedureAddress`
- ❑ The nested callee (each callee becomes a caller for its next callee)

Within each callee

- Pushes preserved registers (`$s0 - $s8`), argument registers (`$a0 - $a1`) onto stack if changed within callee
- Pushes temporary registers (`$t0 - $t9`) onto stack if changed within callee and needed after the call
- Pushes `$ra` for its caller into stack
- Performs the recursive procedure by `jal nestedProcedureAddress`

After returning to each caller

- Pops the preserved registers, argument registers, and temporary registers from stack if there is any
- Pops its `$ra`
- Puts return results in `$v0` - `$v1`
- Invokes `jr $ra` to go back to the caller

Question 1: Translate the following C++ recursive function into a MIPS recursive function.

```
int multiply(int p1, int p2) {  
    if (p2 == 0)  
        return 0;  
    return p1 + multiply(p1, p2 - 1);  
}
```

Question 2: Translate the following C++ recursive function into a MIPS recursive function.

```
int fact(int p) {  
    if (p < 1)  
        return 1;  
    else  
        return (p * fact(p-1));  
}
```

Question 3: Translate the following C++ recursive function into a MIPS recursive function.

```
int fib(int n) {  
    if (n == 0)  
        return 0;  
    if (n == 1)  
        return 1;  
    return (fib(n-1) + fib(n-2));  
}
```


MIPS recursion

Recursive procedures
- examples

Exercises

Exercise 1: Translate the following C++ recursive function into a MIPS recursive function.

```
int sum(int x) {  
    if (x == 0)  
        return 0;  
    return x + sum(x - 1);  
}
```