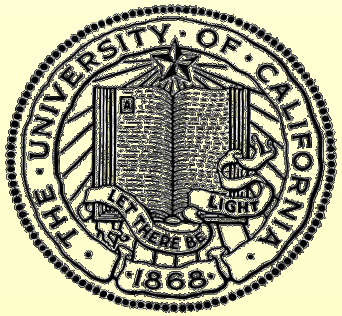
Functions



Final Exam

- Wednesday, March 17 8:00–11:00 a.m.
  - Registrar's Office
- Exam available on Canvas from **8 am till 11 am** local time (Santa Cruz)
- Exam duration < 90 minutes (will confirm this in a later Piazza post before exam)
- Syllabus: ALL lectures (comprehensive)

Functions

- What are functions?
  - Same as subroutines, but can pass arguments and return values
  - E.g. C function definition:
    - int squared(int a) { return(a*a); }
  - E.g. C function call:
    - ret=squared(b);
- Syscall was not really a function…
  - Argument $a0
  - Type of syscall $v0
  - Return value $v0

Functions in Assembly

- Implemented as subroutine calls in assembly
  - ret=squared(b);
    - Load variable (“b”) from memory into register argument
    - Jump to subroutine label “squared”
    - Put result in register for return
    - Return from subroutine call
    - Store result into variable (“ret”) in memory
- **What happens to the registers I am using?**
- **How do I get back from the subroutine?**

The responsibility of saving registers

- MIPS splits the duties into two sets of registers
  - $t0-$t9 are **"caller save"** where the calling function must save them before calling a function.
  - $s0-$s7, $ra are **"callee save"** where a called function must save them and restore them before returning.
    - Note jal/jalr to call functions "uses" the $ra register

Functions Arguments and Return values

- Four arguments $a0-$a3

- Two return values $v0-$v1

Caller Duties

- If it was using $t0-$t9 to store values…
  - subroutine could also use these without saving them!
  - **Caller must save $t0-$t9** if it wants to preserve them during function call
  - These are "caller save" registers
- Calls a function with jal

Callee Duties

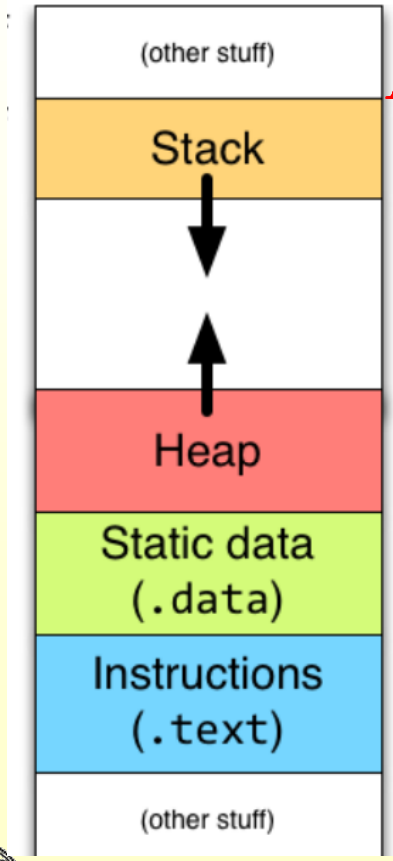
- Can use $t0..$t9 as it wants without saving/restoring
- **Must save/restore $s0..$s7** if it wants to use them
- Must save/restore $ra if it calls other functions
- Return with jr $ra


```
called_function      :
        # Save if use $s0..$s7, $ra

        # Put return value in $v0..$v1
        # Restore $s0..$s7, $ra
        jr   $ra
```

Heap Memory

- The heap is a memory used by programming languages to store global variables.
- By default, all global variables are stored in heap memory space.
- It supports Dynamic memory allocation.
- The heap is not managed automatically for you and is not as tightly managed by the CPU. It is more like a free-floating region of memory.



local variables, return addresses
RW

dynamically allocated memory like
Java objects
RW

global data
(initialized when process starts)
RW

assembled code
(initialized when process starts)
RX

The end..

- END OF LECTURES!
- ALL THE BEST FOR YOUR FINALS!
- Special thanks to the TAs and Tutors for helping in class schedule