Recursion:

Definition:

Recursion: If you still don't get it, see: "Recursion".

Today's lecture

- Recursion
  - My turn (Example)
  - Your turn
- Callee-saved Registers
  - My turn again
- ASCII & NULL-terminated strings
- 2-dimensional arrays in C

Recursion in MIPS

- Last time we talked about function calls…
  - Recursion is just a special case of function calls
- Two parts:
  - Base case: no recursion, often doesn’t call any functions
  - Recursive body: calls itself

Recursion in MIPS (single function call)

Suggestions for implementing recursive function calls in MIPS

1. Handle the base case first
   - Before you allocate a stack frame if possible
2. Allocate stack frame
3. Save return address
4. Recursive Body:
   a) Save any registers needed after the call
   b) Compute arguments
   c) Call function
   d) Restore any registers needed after the call
   e) Consume return value (if any)
5. Deallocate stack frame and return.

Recursion in MIPS (multiple calls – caller save)

Suggestions for implementing recursive function calls in MIPS

1. Handle the base case first
   - Before you allocate a stack frame if possible
2. Allocate stack frame
3. Save return address
4. For each function call: *(suggestion: use **$s** registers if >1 call)*
   a) Save any registers needed after the call
   b) Compute arguments
   c) Call function
   d) Restore any registers needed after the call
   e) Consume return value (if any)
5. Deallocate stack frame and return.

Recursion in MIPS (multiple calls – callee save)

Suggestions for implementing recursive function calls in MIPS

1. Handle the base case first
   - Before you allocate a stack frame if possible
2. Allocate stack frame
3. Save return address
4. Save enough **$s** registers to hold your local variables
5. Copy your local variables to **$s** registers
6. For each function call:
   a) ~~Save any registers needed after the call~~
   b) Compute arguments
   c) Call function
   d) ~~Restore any registers needed after the call~~
   e) Consume return value (if any)
7. Restore **$s** registers
8. Deallocate stack frame and return.