CS271 WEEK 5 TUES LECTURE + THURS

Modularization

- Breaking a program into segments that can be solved independent.
- Main procedure becomes just a caller of functions / Processes.

Processes

- Pushes next instruction offset ~~to EIP~~ when Call called, to stack
- Popped by "ret" into EIP.

Programming Process w/ Procedures

- write main and procedure Stubs
  → test!
- declare variables
  → test!
- Implement procedures 1 at a time
  → test each separately
  → Impement : Out, In, Proc...
- Clean up.

System Stack

- Stack is Data Structure
- LIFO ≡ FILO
- Operations: push, pop
- Managed by CPU,
- Uses registers ESP (top of stack) and SS (stack segment)

Pop and Push

- Push decrements ESP by $\overset{\text{for 32-bit}}{4}$, copies value into mem location where ESP points to.

[reg] = value of register @ address

- Pop copies value pointed to by ESP into DEST.
  → increments ESB by 4

Documenting a Procedure

- Description
- Recieves
- Returns
- Preconditions
- Registers modified

Parameters & Arguments

- "Argument" (Actual Parameter) is the value that is being passed.
- "Parameter" (Formal Parameter) is the value recieved by the proc

Parameter Passing

- Input Parameter ≅ "value" or "pass by value"
- Output Parameter ≏ "reference" or "pass by reference". Passed using an address (using offset). Used only as return space for output.
- I/O Parameter = pass by ref, something that is both used and returned by Process
  or edited/modified.

Ways of Passing Parameters

→ Global Vars → Registers → Stack.

- Global Variable Parameters
  - → Bad Idea
- Register Passing Parameters
  - → Bad Idea, but used quire often. Very simple.
- Passing on System Stack
  - → push then pop to reg
  - → Stack frames
  - → Better for system resources