Stacks

October 13

CSC201 Section 002
Fall, 2000
Stack Uses

• Saving and restoring register values
• Saving return addresses for procedure calls
• Passing parameters to/from procedures
• Allocating memory dynamically
Stack Access

- Access is last-in-first-out (LIFO)
  - One use: reversing the order of an array of elements
  - Example: digits produced during base conversion

- Use "push" to store onto stack

- Use "pop" to retrieve and remove from stack
Stack Organization

- The stack location in memory
  - allocated with the "stack" directive
- Direction of stack growth is "upwards" (towards lower-numbered addresses)
- The ESP register points to the "top" of stack
  - "points to" means "contains the address of"
  - initially, just below the start of the stack (i.e., there is no top)
  - this initialization is done by the linker/loader automatically
- The EBP register is also used to point to a portion of the stack
  - Usually, the location of subroutine parameters
The Push Operation

- Instruction: "push operand"
- First, decrements ESP by 4
- Then copies operand (doubleword or byte) to m(ESP)
  - This is a form of (implicit) register indirect addressing
- "operand" may be memory or register or immediate value
Example

Stack area in memory

Empty

after “push EAX”
The POP Operation

- Instruction: "pop operand"
- First, copies doubleword (or byte) at m(ESP) into operand
- Then increments ESP by 4
Example

After 2 pushes

after "pop oper1"

SP $\rightarrow$

EBX

EAX

SP $\rightarrow$

EAX
Details (PUSH and POP)

- No effect on flags
- Little-endian ordering of bytes within a doubleword on the stack
- There must be a POP for every PUSH
  - Make sure they are in reverse order!
"Empty" and "Full" Stacks

• Empty stack: ESP points just below the first element of the stack
  - POP is an error on empty stack; check first!

• Full stack: ESP points to the top (lowest-numbered) storage location allocated for the stack
  - PUSH is an error on a full stack; check first!
Examples of Stack Use

.stack 1000h ; 4096 bytes =
    ; 1024 doublewords

... push EAX
push 0FF2233AAh
push myvar1

... pop EBX
pop [dword myvar4]
pop EAX
Example

after 3 pushes

after first pop
Checking for Full and Empty

Bottom_of_stack dd ?
Top_of_stack dd ?

...  
  mov  bottom_of_stack, ESP
  mov  top_of_stack, ESP
  iadd  top_of_stack, 1000h
  ...
  compare  ESP, bottom_of_stack
  bgez  stack_empty_dont_pop
  ...


Checking (cont.)

... compare ESP, top_of_stack
      blez stack_full_dont_push
      ...

Dynamic Memory Allocation

- Goal: set aside "n" bytes of storage for temporary use...
- First copy ESP into EBP
- Then subtract "n" from ESP
- Refer to (I.e., address) an element in this area of memory using...
  - based-indexed (or based-displacement) addressing mode
  - with EBP as the base register
  - displacement will be negative!
Dynamic Allocation Example

mov EBP, ESP

isub ESP, num_bytes_to_allocate
Before dynamic allocation  after dynamic allocation
Warning!

• After "popping" an element from the stack, don't count on it's value still being there later!
Some Clarifications

• Push
  - Can push a byte-length immediate value
  - (Can push from a word-length register, or a word from memory, or a word-length immediate value)
  - Can push from a double-word length register, or a double-word from memory, or a double-word length immediate value

• Pop
  - Cannot pop byte-length operands
  - (Can pop to a word-length register, or a word to memory)
  - Can pop to a double-word length register, or a double-word to memory