Lecture 17: Recursion and Shift/Rotate

- Recursion
- Shift
- Rotate
- Homework 4

Recursion

- Recursion is when an algorithm is defined in terms of itself.
- Example: Factorial

  n! = n * (n-1)*(n-2)*(n-3)…(1)
  fact(0) = 1

  4! = 4 * 3 * 2 * 1 = 24

- Defined in terms of itself:
  fact(n) = n * fact(n-1)
  fact(0) = 1

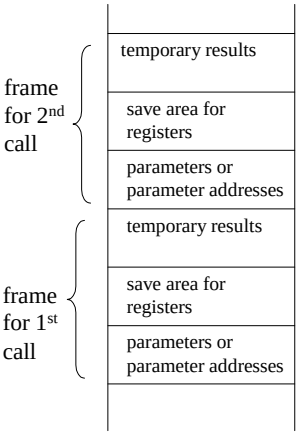
Issues with Recursion

- Implementation
- Parameter Preservation
- Variables

Solution

- Parameters, registers, and temporary results need to be stored in a different place in memory for each invocation of the recursive function.
- How?

Stack Frames in Recursion



data pushed on the stack can be retrieved in reverse order as the function returns from its nesting.

BP can be used to permit access to items in a frame.

```
        main proc
0000            mov     ax, 3       ;calculate 3!
0003            push    ax
0004            call    Factorial
0007            mov     ax, 4c00h
000A            int     21h
        main endp

        Factorial proc
000C            push    bp
000D            mov     bp, sp
000F            mov     ax, [bp+4]  ;get n
0012            cmp     ax, 1       ;n <= 1?
0015            ja      L1          ;no: continue
0017            mov     ax, 1       ;yes: return 1
001A            jmp     L2
001D    L1:     dec     ax
001E            push    ax          ;Factorial(n-1)
001F            call    Factorial
0022            mov     bx, [bp+4]  ;get n
0025            mul     bx          ;AX=AX*BX
0027    L2:     pop     bp
0028            ret     2           ;AX holds result
        Factorial  endp
```

Shift and Rotate

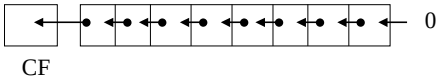
- Shift and rotate instructions provide a way to move bits around in an operand.
  - SHL shift left
  - SHLD double-precision shift left
  - SHR shift right
  - SHRD double-precision shift right
  - SAL shift arithmetic left
  - SAR shift arithmetic right
  - ROL rotate left
  - ROR rotate right
  - RCL rotate carry left
  - RCR rotate carry right

SHL – shift left

- Each bit in the destination operand is shifted to the left, filling the lowest bit with zero.
- The high bit is moved to the carry bit, the bit that was in the carry bit is discarded.

```
SHL dest,1        ;1 bit to left
SHL dest, CL      ;CL holds # of bits
SHL dest, immed8 ; shift immed8
                  ;# of bits
```

SHL – continued




```
shl bl, 1  ;shift bl 1 bit to the left
                ;bl = 05h, new bl =

shl wordval, 1 ;16-bit memory operand
                ;wordval = 50A7h, new
                ;wordval =

shl al, cl ;shift using count in cl
                ;al = 4Bh, cl = 4, new al =

shl bx, 5 ;shift left 5
                ;bx = 50A7h, new bx =
```

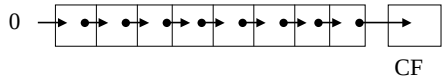
Uses for SHL

- The most common use is high-speed multiplication.
  - mov dl, 1 ;dl = 1
  - shl dl, 1 ;dl = 2
  - shl dl, 1 ;dl = 4
  - shl dl, 1 ;dl = 8
  - etc.
- Each shift left multiplies by a power of 2!
- Shifting is much faster than multiplication.

SHR – shift right

- Each bit in the destination operand is shifted to the right, replacing the highest bit with a zero.
- The low bit is copied into the carry flag, and the bit that was in the carry flag is lost.
- Instruction formats are the same as in SHL.

SHR - continued




```
shr bl, 1  ;shift bl 1 bit to the right
                ;bl = 05h, new bl =

shr wordval, 1 ;16-bit memory operand
                ;wordval = 50A7h, new
                ;wordval =

shr al, cl ;shift using count in cl
                ;al = 4Bh, cl = 4, new al =

shr bx, 5 ;shift right 5
                ;bx = 50A7h, new bx =
```

Uses for SHR

- You can use SHR to divide a number by 2:

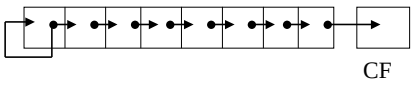
```
mov dl, 32 ;00100000b
shr   dl, 1          ;00010000b (dl = 16)
```

- You can use multiple shifts to divide by larger powers of 2:
  - mov al, 01000000b ;al = 64
  - shr al, 3 ;divide by 8
    ; al = 00001000b = 8
- what numbers *can't* you divide this way? -> signed numbers! (sign is lost)

SAR

- So how do you divide signed numbers by two? SAR and SAL (shift arithmetic right and shift arithmetic left).
- SAL is identical to SHL and is included in the instruction set for completeness
- SAR shifts each bit to the right and makes a copy of the sign bit, preserving the sign of the number.

SAR – cont.



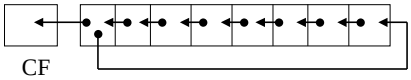

```
mov al, 0F0h      ;al = 11110000b
                  ;(-16)
sar  al, 1        ;al =


mov  dx, 8000h    ;dx =
                  ;1000000000000000b
                  ;(-32768)
sar  dx, 5        ;dx =
```

shifting right 5 times is the same as dividing by 2**5.

ROL

- ROL, rotate left, moves each bit to the left. The highest bit is copied into the carry flag and the lowest bit.
- In rotate instructions, bits are never lost – as they rotate off one end, they rotate back on to the other.

ROL – cont.

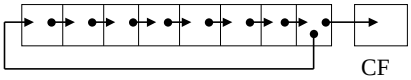



```
mov al, 40h     ;al = 01000000b
rol     al, 1           ;al =

rol     al, 1           ;al =

rol     al, 1           ;al =
```

Example – Using ROL

- Example 1, p233 of Irvine

ROR



- Like ROL, except rotating right.

```
mov al, 01h  ;al =

ror   al, 1           ;al =

ror     al, 1         ;al =
```