CIS21JA - Intro to x86 Processor Assembly Study Guide

Connor Petri

March 4, 2025

Data and Math

Data Instructions

M - Memory, R - Register, C - Constant

mov {MR-}, {MRC} Move data from op2 to op1

cbw Convert byte in AL to word in AX by sign-extending the most significant bit of AL

cwd Convert word in AX to doubleword in DX:AX by sign-extending the most significant bit of AX

cdq Convert doubleword in EAX to quadword in EDX:EAX by sign-extending the most significant bit of EAX

Math Instructions

M - Memory, R - Register, C - Constant

add {MR-}, {MRC} Add two values and store in op1

sub {MR-}, {MRC} Subtract two values and store in op1

mul {MR-} Multiply eax by op1 and store in edx:eax

div {MR-} Divide edx:eax by op1, quotient in eax, remainder in edx

imul {MR-} Signed multiply eax by op1 and store in edx:eax

idiv {MR-} Signed divide edx:eax by op1, quotient in eax, remainder in edx

Bitwise Instructions

M - Memory, R - Register, C - Constant

and {MR-}, {MRC} Bitwise AND two values and store in op1

or {MR-}, {MRC} Bitwise OR two values and store in op1

xor {MR-}, {MRC} Bitwise XOR two values and store in op1

not {MR-} Bitwise NOT op1

shl {MR-}, {MRC} Shift op1 left by op2 bits

shr {MR-}, {MRC} Shift op1 right by op2 bits with zero extension

sal {MR-}, {MRC} Same as shl

sar {MR-}, {MRC} Shift op1 right by op2 bits with sign extension

rcl {MR-}, {MRC} Rotate bits of op1 left through the carry flag by op2 bits

rcr {MR-}, {MRC} Rotate bits of op1 right through the carry flag by op2 bits

The Stack

Definition The stack is a region of memory that is used to store data temporarily. It is a LIFO (Last In, First Out) data structure. The stack is used to store local variables, function arguments, and return addresses. The stack pointer register `esp` points to the top of the stack. The stack grows downward in memory.

Instructions

push {Mem/Reg/Literal} Pushes a value onto the stack

pop {Mem/Reg/Literal} Pops a value from the stack

Passing arguements using the stack

Setting up stack frame The stack frame register `ebp` is used to point to the base of the current stack frame. This is used to access local variables and function arguments explicitly.

```
foo proc

    push ebp        ; save the old base pointer
    mov ebp, esp    ; set the base pointer to the
                    ; current stack pointer

    mov eax, [ebp+8] ; access the first argument
    mov ebx, [ebp+12] ; access the second argument

    pop ebp         ; restore the old base pointer
    ret 12          ; return and clean up the stack (12 bytes)
                    ; (8 bytes for the arguments,
                    ; 4 bytes for return address)

foo endp
```