**What is kernel mode and user mode? (You must know this for CSCI 435!)**

kernel mode is intended to run the operating system and allows all instruction to be executed.

user mode is intended to run application programs and does not permit ceratin sensitive instructions to be executed.

**Expanding opcodes**

An expanding opcode is an opcode that can expand its length taking space when need it where usually the address are hold. This proccess in a 16 bit instruction gives the possibility to have 15 three addresses insruction, 14 2 addresses instruction 31 one addresses instruction and 16 0 addresses instruction

**Prefix opcodes**

Are opcodes that have bytes infront of the instruction in order to change is action.

**Operands, type and addressing**

Operands are entities operated upon by the instruction

Instructions work on sources of data called operands.

Here are the Operand Types and corresponding Addressing Modes:

Register - an 8-bit or 16-bit register on the 808680486, 32-bit on the 80386/486/Pentium.

Immediate - a constant value contained in the instruction itself.

Direct memory - a fixed location in memory.

Indirect memory - a memory location determined at run time by using the address stored in one or two registers.

types are the type of instruction is going to be performed on the opcode

addressing is the location of where the operands are located

Common Addressing Modes.

**Immediate Addressing.**

The operand it self is contained in the address. The operand is called imidiate operand because it is automatically fetched from memory at the same time the instruction itself is fetched.

- has the virtue of not requirinf an extra fetch in memory

-disdvatages using small integer constants

**Direct Addressing (Absolute Addressing).**

memory address encoded directly in the instruction

value can change, but address determined at compile/assemble time

can only really be used for global variables who address is known ar compile time

**Register Addressing.**

.k.a., register mode

like direct addressing, but instead of the instruction containing a memory address, it contains a register address (i.e., register number)

Many of the previous examples used register addressing

MOV R1,R4

this is the most common type of addressing

in load/store architectures (e.g., UltraSPARC II) all instructions use this mode exclusively (except the load and store instructions)

**Register Indirect Addressing (Indirect Addressing).**

the memory address of the operand is stored in a register, and the register number is stored in the instruction. An address used this way is called a pointer.

**Indexed Addressing**.

Address is given as an offset (displacement) to an address held in a register

this is exactly what we used to reference local variables and parameters in IJVM where the address was in the LV register.

You can even invert this, where the memory location is in the instruction, and the offset is in a register

**Implied Addressing.**

Implied addressing is an addressing mode which specifies no address at all

**Offset or Relative Addressing.**

Relative addressing is the addressing mode used by all conditional-branch instructions in the 65xx instruction set: Beyond the opcode for the instruction itself, all these instructions take up a single, signed-integer byte that specifies, in relative terms, how far "up" or "down" to jump if the required conditions are met: This signed 8-bit figure is called the offset.

**Scale, index, base.**

Is an additional byte that specifies a scale factor as well as two registers.

Branching

Unconditional

It results in always brancing.

Conditional

Is if a condition is met then branch to a particular memory address to execute the instruction.

Subroutine and return (link in mips, $ra register)

Interrupt

Implementation of conditional branching, flag registers, register set and test