

# Addressing Modes

October 6

CSC201 Section 002

Fall, 2000

# Register Direct Mode (Pentium: Register Indirect Mode)

- Idea: a register contains the \*address\* in memory of the operand
  - basically, use a register as a pointer, instead of using a memory variable
- Additional memory accesses: 1
  - ( must have previously loaded the address into the register)
- Bits in instruction: enough for register ID
- Example:
  - `la reg3, myvar`
  - `add [reg3], 3`
- vs.
  - `la addr1, myvar`
  - `add m(addr1), 3`

# Relative Mode (Pentium: Based-Displacement)

- Idea: a register contains an \*address\*; add a "small" constant to get the effective address of the operand
- Additional memory accesses: 1
- Bits in instruction: enough for register ID + small constant
- Example:
  - `la reg3, myarray`
  - `add 4[reg3], 3`
- Alternative syntax:
  - `la reg3, myarray`
  - `add [reg3+4], 3`

# Indexed Mode

## (Pentium: Based-Indexed Mode)

- Idea: one register contains an \*address\*, add value in another register to get the effective address of the operand
- Additional memory accesses: 1
- Bits in instruction: enough for two register IDs
- Example:
  - la reg1, myarray
  - la reg2, 4
  - add [reg1+reg2], 3
- Alternative syntax
  - add [reg1][reg2], 3

# Pentium: Based-Indexed with Displacement

- Suitable for 2-D array access
- Skipping this one...

# PC Relative

- Modification to based-displacement; the base register implicitly = the program counter (PC)
  - PC contains address of the \*NEXT\* instruction to execute
  - so, branch to a location some displacement away from next instruction
  - sole benefit: displacements are smaller (fewer bits) than absolute addresses
- Additional memory accesses: 0
- Bits in instruction: the small constant (displacement)
- Example:
  - `br my_label ; "my_label" is replaced by the assembler with the displacement to the target address`

# Indirect (Pentium: not available)

- Idea: an address is given; at this memory location is the actual (effective) memory address of the operand
  - standard use of pointers: linked lists, arrays, etc.
- Additional memory accesses: 2
- Bits in instruction: memory address (32 bits)
- Example:
  - `la`      `addr1, myvar`
  - `add`     `m(addr1), 3`

# Code Examples (C and SASM)

- `Int d1[10];`
- `D1[3] = 5;`
- Use indirect mode / register indirect mode
- Use based-displacement mode
- Use based-indexed mode



# Code Examples (C and SASM)

- `Int d1[10][6];`
- `d1[3][4] = 5;`
- Use indirect mode / register indirect mode
- Use based-displacement mode
- Use based-indexed mode

# Code Examples (C and SASM)

- Struct {
  - Int age;
  - Char gender;} s1;
- S1.gender = 'f';
- Use indirect mode / register indirect mode
- Use based-displacement mode
- Use based-indexed mode

# Code Examples (C and SASM)

- Struct {
  - Int age;
  - Char gender;} s1[10];
- S1[5].gender = 'f';
- Use indirect mode / register indirect mode
- Use based-displacement mode
- Use based-indexed mode