LECTURE THREE

INSTRUCTION SET ARCHITECTURE

INSTRUCTION TYPES

- The following are the categories of instructions:
  - Data movement
  - Arithmetic
  - Boolean
  - Bit manipulation (shift and rotate)
  - I/O
  - Transfer of control
  - Special purpose

INSTRUCTION TYPES cont…

- The data transfer instruction must specify several things:
  - First, the location of the source and destination operands must be specified. Each location could be memory, a register, or the top of the stack.
  - Second, the length of data to be transferred must be indicated.
  - Third, the mode of addressing for each operand must be specified.

INSTRUCTION TYPES cont…

- In terms of processor action, data transfer operations are perhaps the simplest type.
- If both source and destination are registers, then the processor simply causes data to be transferred from one register to another; this is an operation internal to the processor.

INSTRUCTION TYPES cont…

- If one or both operands are in memory, then the processor must perform some or all of the following actions:
  1. Calculate the memory address, based on the address mode
  2. If the address refers to virtual memory, translate from virtual to real memory address.
  3. Determine whether the addressed item is in cache.
  4. If not, issue a command to the memory module.

Common Data Movement Operations

Data movement operation	Meaning
MOVE	Move data (a word or a block) from a given source (a register or a memory) to a given destination
LOAD	Load data from memory to a register
STORE	Store data into memory from a register
PUSH	Store data from a register to stack
POP	Retrieve data from stack into a register

INSTRUCTION TYPES cont…

- ***Arithmetic operations*** include those instructions that use integers and floating point numbers.
- Many instruction sets provide different arithmetic instructions for various data sizes.

Common Arithmetic Operations

Arithmetic operations	Meaning
ADD	Perform the arithmetic sum of two operands
SUBTRACT	Perform the arithmetic difference of two operands
MULTIPLY	Perform the product of two operands
DIVIDE	Perform the division of two operands
INCREMENT	Add one to the contents of a register
DECREMENT	Subtract one from the contents of a register

INSTRUCTION TYPES cont…

- ***Boolean logic instructions*** perform Boolean operations, much in the same way that arithmetic operations work.
- There are typically instructions for performing AND, NOT, and often OR and XOR operations.

Common Logical Operations

Logical operation	Meaning
AND	Perform the logical ANDing of two operands
OR	Perform the logical ORing of two operands
EXOR	Perform the XORing of two operands
NOT	Perform the complement of an operand
COMPARE	Perform logical comparison of two operands and set flag accordingly
SHIFT	Perform logical shift (right or left) of the content of a register
ROTATE	Perform logical shift (right or left) with wraparound of the content of a register

INSTRUCTION TYPES cont…

- The **arithmetic shift** operation treats the data as a signed integer and does not shift the sign bit.
  - On a right arithmetic shift, the sign bit is replicated into the bit position to its right.
  - On a left arithmetic shift, a logical left shift is performed on all bits but the sign bit, which is retained.
- **Rotate**, or cyclic shift, operations preserve all of the bits being operated on.
  - One use of a rotate is to bring each bit successively into the leftmost bit, where it can be identified by testing the sign of the data (treated as a number).

INSTRUCTION TYPES cont…

Input	Operation	Result
10100110	Logical right shift (3 bits)	00010100
10100110	Logical left shift (3 bits)	00110000
10100110	Arithmetic right shift (3 bits)	11110100
10100110	Arithmetic left shift (3 bits)	10110000
10100110	Right rotate (3 bits)	11010100
10100110	Left rotate (3 bits)	00110101

INSTRUCTION TYPES cont…

- ***Bit manipulation instructions*** are used for setting and resetting individual bits (or sometimes groups of bits) within a given data word.
- These include both arithmetic and logical shift instructions and rotate instructions, both to the left and to the right.

INSTRUCTION TYPES cont…

- ***Input and output instructions (I/O instructions)*** are used to transfer data between the computer and peripheral devices.
- The two basic I/O instructions used are the INPUT and OUTPUT instructions.
- INPUT instruction is used to transfer data from an input device to the processor.
- Examples of input devices include a keyboard or a mouse.

INSTRUCTION TYPES cont…

- Input devices are interfaced with a computer through dedicated input ports.
- Computers can use dedicated addresses to address these ports.
- Suppose that the input port through which a keyboard is connected to a computer carries the unique address 1000.

INSTRUCTION TYPES cont…

- Execution of the instruction INPUT 1000 will cause the data stored in a specific register in the interface between the keyboard and the computer, call it the input data register, to be moved into a specific register (called the accumulator) in the computer.

INSTRUCTION TYPES cont…

- Similarly, the execution of the instruction OUTPUT 2000 causes the data stored in the accumulator to be moved to the data output register in the output device whose address is 2000.

INSTRUCTION TYPES cont…

- Alternatively, the computer can address these ports in the usual way of addressing memory locations.
- In this case, the computer can input data from an input device by executing an instruction such as ***MOVE Rin, R0***.
- This instruction moves the content of the register *Rin* into the register *R0*.

INSTRUCTION TYPES cont…

- Similarly, the instruction ***MOVE R_0, Rin*** moves the contents of register *R0* into the register *Rin*, that is, performs an output operation.

INSTRUCTION TYPES cont…

- ***Control (sequencing) instructions*** are used to change the sequence in which instructions are executed.
- They take the form of CONDITIONAL BRANCHING (CONDITIONAL JUMP), UNCONDITIONAL BRANCHING (JUMP), or CALL instructions.
- A common characteristic among these instructions is that their execution changes the program counter (PC) value.

INSTRUCTION TYPES cont…

- The change made in the PC value can be unconditional, for example, in the unconditional branching or the jump instructions.
- In this case, the earlier value of the PC is lost and execution of the program starts at a new value specified by the instruction.

INSTRUCTION TYPES cont…

- The change made in the PC by the branching instruction can be conditional based on the value of a specific flag.
- Examples of these flags include the *Negative (N), Zero (Z), Overflow (V),* and *Carry (C).*
- These flags represent the individual bits of a specific register, called the **CONDITION CODE (CC) REGISTER**.
- The values of flags are set based on the results of executing different instruction

Examples of Condition Flags

Flag name	Meaning
Negative (N)	Set to 1 if the result of the most recent operation is negative, it is 0 otherwise
Zero (Z)	Set to 1 if the result of the most recent operation is 0, it is 0 otherwise
Overflow (V)	Set to 1 if the result of the most recent operation causes an overflow, it is 0 otherwise
Carry (C)	Set to 1 if the most recent operation results in a carry, it is 0 otherwise

Transfer of Control Operations

Transfer of control operation	Meaning
BRANCH-IF-CONDITION	Transfer of control to a new address if condition is true
JUMP	Unconditional transfer of control
CALL	Transfer of control to a subroutine
RETURN	Transfer of control to the caller routine

ADDRESSING

- There are two most important addressing issues: the **types of data** that can be addressed and the various **addressing modes.**

Data Types

- **Numeric data** consists of integers and floating point values.
  - *Integers* can be signed or unsigned and can be declared in various lengths.

    For example, in C++ integers can be *short* (16 bits), *int* (the word size of the given architecture), or *long* (32 bits).
  - *Floating point numbers* have lengths of 32, 64, or 128 bits.

Data Types cont…

- **Nonnumeric data types** consist of strings, Booleans, and pointers.
  - *String instructions* typically include operations such as copy, move, search, or modify.
  - *Boolean operations* include AND, OR, XOR, and NOT.
  - *Pointers* are actually addresses in memory.

Address Modes

- Addressing modes allow us to specify where the instruction operands are located.
- An addressing mode can specify a constant, a register, or a location in memory.
- Certain modes allow shorter addresses and some allow us to determine the location of the actual operand, often called the *effective address of* the operand, dynamically.

Basic Addressing Modes

Immediate Addressing : the data to be operated on is part of the instruction.

- For example, if the instruction is **Load 008,** the numeric value 8 is loaded into the AC.
- The 12 bits of the operand field do not specify an address— they specify the actual operand the instruction requires.

Basic Addressing Modes cont…

- **Advantage:** *no memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle.*
- **Disadvantage:** *is that the size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length.*

Basic Addressing Modes cont…

Direct Addressing: the value to be referenced is obtained by specifying its memory address directly in the instruction.

- For example, the instruction is **Load 008,** the data value found at memory address 008 is loaded into the AC.
- Is typically quite fast because, although the value to be loaded is not included in the instruction, it is quickly accessible.

Basic Addressing Modes cont…

- It is also much more flexible than immediate addressing because the value to be loaded is whatever is found at the given address, which may be variable.
- The obvious limitation is that it provides only a limited address space.

Register Addressing :a register, instead of memory, is used to specify the operand.

- This is very similar to direct addressing, except that instead of a memory address, the address field contains a register reference.
- The contents of that register are used as the operand.

Basic Addressing Modes cont…

Indirect Addressing: the bits in the address field specify a memory address that is to be used as a pointer.

- The effective address of the operand is found by going to this memory address.
- For example, the instruction is **Load 008**, the data value found at memory address 008 is actually the effective address of the desired operand.

Basic Addressing Modes cont…

- Suppose we find the value 2A0 stored in location 008.
- 2A0 is the “real” address of the value we want.
- The value found at location 2A0 is then loaded into the AC.

Basic Addressing Modes cont…

- In a variation on this scheme, the operand bits specify a register instead of a memory address, this mode is called ***register indirect addressing,*** works exactly the same way as indirect addressing mode, except it uses a register instead of a memory address to point to the data.
- For example, if the instruction is **Load R1,** we would find the effective address of the desired operand in R1.

Basic Addressing Modes cont…

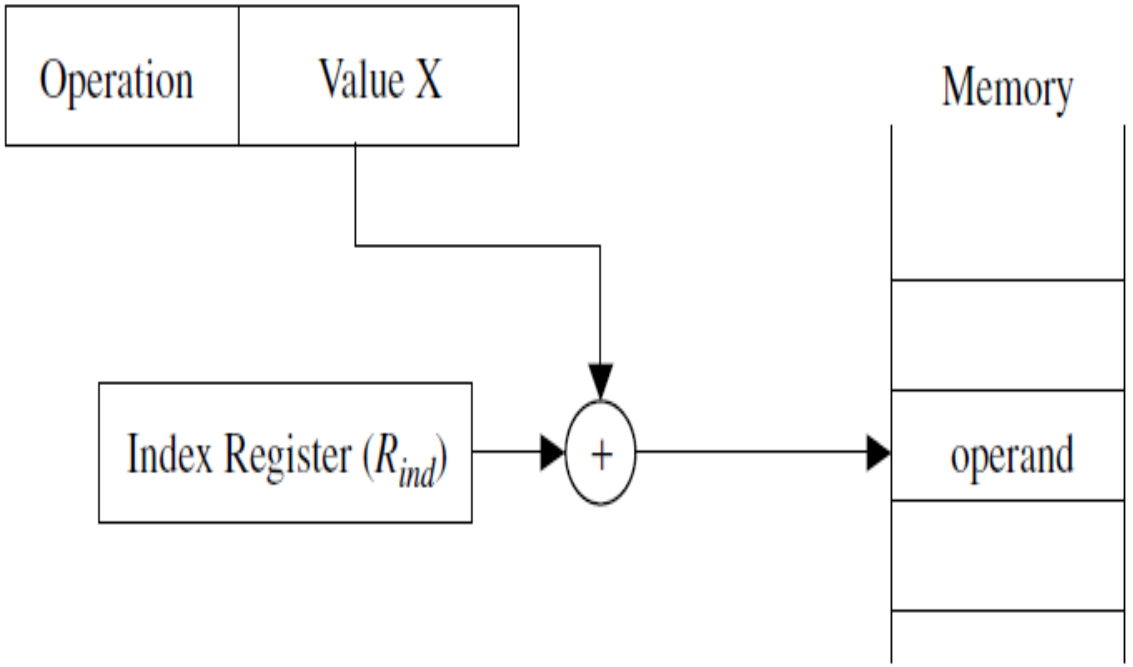
Index addressing mode, the address of the operand is obtained by adding a constant to the content of a register, called the index register.

- Consider, for example, the instruction *LOAD X(Rind), Ri*.
- This instruction loads register *Ri* with the contents of the memory location whose address is the sum of the contents of register *Rind* and the value *X*.

Basic Addressing Modes cont…

- Index addressing is indicated in the instruction by including the name of the index register in parentheses and using the symbol X to indicate the constant to be added.

Illustration of the indexed addressing mode



Example

- Suppose we have the instruction **Load 800,** and the memory and register R1 shown:

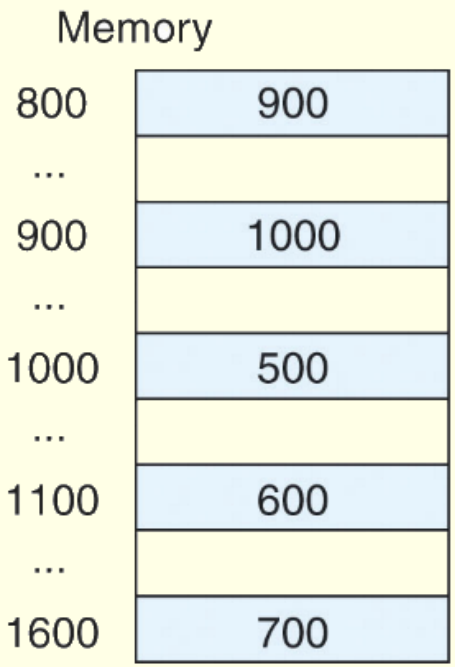
Memory

800	900
...	
900	1000
...	
1000	500
...	
1100	600
...	
1600	700

R1 | 800

LOAD 800

Mode	Value Loaded into AC
Immediate	
Direct	
Indirect	
Indexed	



R1 | 800

LOAD 800

Mode	Value Loaded into AC
Immediate	800
Direct	900
Indirect	1000
Indexed	700

Tutorial

- MIPS ARCHITECTURES
- PROGRAMMING EXAMPLES (From the book "fundamentals of computer organization")