Instruction Set Architecture

Most of this material is covered in chapter 2 of optional text book “Computer Organization and Embedded Systems” by Hamacher, Vranesic, Zaky, and Manjikian.

# Memory Locations and Addresses

## Byte Addressability

Byte-addressable memory means successive addresses refer to successive byte locations.

This for example, if the word length for a machine is 32 bits, successive words are located at addresses 0, 4, 8, ……, with each word consisting of 4 bytes.

## Big-Endian and Little Endian Byte Ordering

Big-Endian : Lower byte addresses are used for the more significant bytes.

Little-Endian : Lower byte addresses used for the less significant bytes

## Word Alignment

Word locations have *aligned* addresses if they begin at a byte address that is a multiple of the number of bytes in a word.

# Instructions and Instruction Sequencing

## Processor Registers

Since the CPU does not have access to directly operate on contents of memory, it contains some special internal memory locations referred to as registers. There are specific instruction for loading data from memory to registers and registers to memory.

The basic types of operations that can be performed by the CPU are…

* Data Transfers between memory and processor registers.
* Arithmetic and logic operations (usually operate on the processor registers rather that directly on memory).
* Program sequencing (branch and conditional branch instructions)
* Control (for example changing mode of operation, status flags, interrupt flags etc)
* I/O transfers, (in some microcontrollers I/O devices may have special instructions to access them). For our purposes using Nios, I/O devices will be memory mapped, i.e. they will be accessed using the same instructions used to access memory.

## Register Transfer Notation

To describe the transfer of information, the contents of any location are denoted by placing square brackets around its name, for example…

R2 ← [LOC]

Means that the contents of location [LOC] are transferred into the processor register R2.

## Assembly-Language Notation

LOAD R2, LOC

Is the assemble language equivalent of R2 ← [LOC]. The actual pneumonic may vary depending on the specific assembly language target, for example LOAD, LD, STA, etc.

## Indirection and Pointers

The indirect addressing mode allows us to use registers in a way that is analogous to using pointers in C code.

If a processor register holds the address of the operand, (the operand is the data we are operating on),then we put the register name in parenthesis.

For Example

MOVE R4, #A26EBB

LOAD R5, (R4)

Will load register R5 from memory address location A26EBB.

## Branching

Instead of executing the next instruction in memory, we branch to another instruction usually located at some offset, (positive or negative) from the branch instruction. Branches are typically conditional, that is to say that the branch away from the normal program flow will only happen if some condition is met.

←→