**2.3 – Operands of the Computer Hardware**

Memory Operands

Arithmetic operations occur only in hardware = registers in MIPS instructions so must include instructions that transfer data between memory and registers

* Data transfer instructions – command that moves memory and registers
* Address – access word in memory

Constant or Immediate Operands

**2.4 – Signed and Unsigned Number**

Two’s complement representation has the advantage that all negative numbers have a 1 in the most significant bit (sign bit)

Signed load = copy sign repeatedly to fill the rest of the register

Unsigned load = fills with 0s to the left of the data, since the number represented by bit pattern is unsigned

Useful shortcuts = negation + sign extension

**2.5 – Representing Instructions in the Computer**

Instructions = series of high and low electronic signals and may be represented as numbers

Today’s computers are built on two key principles:

1. Instructions are represented as numbers
2. Programs are stored in memory to be read or written, just like numbers

**2.6 – Logical Operations**

Which operations can isolate a field in a word?

1. AND
2. A shift left followed by a shift right

**2.7 – Instructions for Making Decisions**

Creating labels and branches avoids burden of writing explicit labels – benefit of writing high-level

Basic block – sequence of instructions without branches

Jump address table – table of address of alternative instruction sequences

The bedrock statement that implements statements for decisions and loops at instruction set level = conditional branch

**2.8 – Supporting Procedures in Computer Hardware**

Procedures are one way to implement abstraction in software

**2.9 – Communicating with People**

ASCII – American Standard Code for Information Interchange 8-bit bytes

**2.10 – MIPS Addressing for 32-bit Immediates and Addresses**

Addressing modes (page 116)

1. **Immediate addressing**
2. **Register addressing**
3. **Base/displacement addressing**
4. PC-relative addressing
5. Pseudodirect addressing

Branch Instruction v. Jump

* Conditional branch instruction must specify two operands in addition to the branch address
* Jump operands involves PC and offset bits
* Branch instruction: program counter = register + branch address

2.11 – Parallelism and Instructions: Synchronization

**2.12 – Translating and Starting a Program**

Step 1 – Compiler

* Transforms C program to assembly language

Step 2 - Assembler

* Symbol table

Step 3 - Linker

* Three steps

1. Place code and data modules symbolically in memory
2. Determine addresses of data and instruction labels
3. Patch both the internal and external references

Step 4 – Loader

**2.13 – A C Sort Example to Put It All Together**

Swapping array elements and sort them

**2.14 – Array versus Pointers**

Array – must have “multiply” and add inside the loop because I is incremented and each address must be recalculated from the new index

Pointer – increments pointer p directly; moves scaling shift and the array bound addition outside the loop – reduces instructions executed per iteration

Pointer Advantages

* Optimization of strength reduction
* Induction variable elimination

**2.16 – Real Stuff: ARMv7 (32-bit) Instructions**

ARM

* More than 9 billion devices in 2011 using ARM
* MIPS has more registers and Arm has more addressing modes
* ARM does not reserve a register to contain 0
* ARM – every instruction has the option of executing conditionally depending on the condition codes
* ARM has support for multiword arithmetic

**2.17 – Real Stuff: x86 Instructions**

Goal to reduce number of instructions executed by program

Danger – cost of simplicity, increases time a program takes to execute because instructions are slower = slower clock cycle time

“golden handcuffs”

2.18 – Real Stuff: ARMv8 (64-bit) Instructions

2.19 – Fallacies and Pitfalls