A Simple Computer Architecture
Program

- The user instructs the computer as to operations to be performed and their sequence by the use of a program
  - Program: a list of instructions that specifies the operations, the operands, and the sequence in which processing is to occur
- Stored Program Concept
  - The program is stored and manipulated in the computer memory just like data
    - Software
- Instruction
  - A collection of bits that instructs the computer to perform a specific operation
The von Neumann Architecture

• The Turing machine (1936)
  • An abstract artifact
    • To analyze the logical foundations of computer systems
• The von Neumann architecture (1945)
  • A practical architecture
  • The conceptual blueprint of almost all computer platforms today
The von Neumann Architecture (contd.)

- Central Processing Unit (CPU) + Memory + Input/Output (I/O) devices
The von Neumann Architecture (contd.)

- **Memory**
  - Holds data items + programming instructions
  - In some computers program and data are stored in separate memory units
    - Data memory and Instruction memory

- **CPU**
  - ALU + set of registers + control unit
    - datapath + control unit

- **I/O devices**
  - LCD monitors, keyboards, printers, mouses, network interface cards, CD-ROMs, and so forth
Machine Instructions

- **Instruction**
  - A collection of bits that instructs the computer to perform a specific operation

- **Executing an instruction**
  - Activating the necessary sequence of microoperations in the data path required to perform the operation specified by the instruction
  - An instruction is *decoded* by the control unit and converted to control signals to the datapath and to the control unit itself
Machine Instructions (contd.)

- Either all the same size (e.g. one 32-bit word for many modern RISC microprocessors) or of different sizes (e.g. for many CISC microprocessors)
  - We assume 32-bit fixed length
- An instruction specifies not only the operation, but also the registers or memory words in which the operands are to be found and the result is to be placed
- Operation code of an instruction (opcode)
  - A group of bits in the instruction that specifies an operation
  - m-bit opcode $\rightarrow 2^m$ different operations
Machine Instructions (contd.)

- **Explicit operands**
  - If the instruction contains special bits for identification of the operand
    - Three binary numbers in an instruction specifying two source registers and one destination register

- **Implicit operands**
  - If it is included as a part of the definition of the operation itself
    - One of the operand is implicitly 1 if we have an increment register instruction
Instruction Set Architecture (ISA)

- Instruction set
  - The collection of instructions for a computer
- Instruction set architecture
  - A thorough description of the instruction set
To execute the instructions in sequence, it is necessary to provide the address in memory of the instruction to be executed:

- The program counter contains the address
- A counter that increments the (instruction word) address by 1
Branching

- Branch instructions modify the program counter to skip over sections of code or to go back to repeat previous code

- Conditional branch instructions
  - Perform a test and branch only if the test is true (1)
    - Testing N, Z, C, and V status signals from the datapath
  - Taken or not-taken

- Unconditional branch instructions (Jumps)
  - Always branch
Assembly Language

- Humans almost never write programs directly in machine language.
- We use an assembly language which is translated by the computer into machine code:
  - A one-to-one correspondence with the resulting machine code instructions.
  - Use of mnemonics for the opcodes.
  - Use of names for:
    - Locations in the program (branch labels).
    - Variables.
    - Constants.
- Examples:
  - ADD R1, R2, R3.
Read-Only Memory (ROM)

- A type of data storage device which is manufactured with fixed contents
  - In its most general sense, the term for any storage system whose contents cannot be altered
  - It is inherently non-volatile storage
    - It retains its contents even when the power is switched off, in contrast to RAM
- ROM is typically used for instruction memory
Control Unit

- We assume that fetching and executing an instruction occurs in a single clock cycle
- Instruction decoder + branch control logic + PC
Control Unit (contd.)

- The instruction decoder is a combinational circuit that provides all of the control words for the datapath, based on the contents of the fields of a machine instruction

  - 32-bit instruction $\rightarrow$ 24-bit control signals

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| DA | AA | BA | BS | FS | DS | RW | MW | PCL | JMP | CND |

- The branch control unit is a combinational circuit

- Control signals for branch control unit
  - CND: for selection of the branch condition status
    - Compared to the N, Z, C, V status signals and then decide whether the branch is taken or not
  - PCL and JMP: for PC
Control Unit (contd.)

- The PC is updated at the beginning of each clock cycle
  - If a jump occurs or a branch is taken in the prior clock cycle, the new PC value is the sum of the previous PC value and the sign-extended address offset
    - Next address = current address + address offset
    - The address offset is encoded in the branch instruction
  - Otherwise, the PC is incremented by 1
Control Unit (contd.)

- Two control signals for PC
  - PCL
    - PC load when PCL = 1
    - PC increment when PCL = 0
  - JMP
    - Jump when JMP = 1 and PCL = 1
    - Conditional branch when JMP = 0 and PCL = 1

- Constant in
  - Instruction[p-1:0]
    - P-bit constant
  - The output from the instruction memory goes to the sign-extend circuit that provides the constant input to the datapath
Control Word Bits for Different Instruction Types

<table>
<thead>
<tr>
<th>Instruction type</th>
<th>BS</th>
<th>DS</th>
<th>RW</th>
<th>MW</th>
<th>PCL</th>
<th>JMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU operation using registers</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Shift operation using registers</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Memory write using register data</td>
<td>0</td>
<td>×</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Memory read using register data</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>ALU operation using a constant</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Shift operation using a constant</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>Conditional branch</td>
<td>×</td>
<td>×</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Unconditional branch (Jump)</td>
<td>×</td>
<td>×</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(×: 0 or 1, either is fine)