Lesson 17:
Encoding of Instructions
Objective

- Learn encoding variable and fixed length encoding
Encoding
Instructions in three address machine

Three-Operand Arithmetic Instruction

- SUB $r_1$, $r_2$, $r_3$  
- Input registers
- Output register
- Implied

Two-Operand Arithmetic MUL Instruction

- MUL $r_1$, $r_2$
- Input registers
- Output register
- Register for extended bits
- Implied

Two-Operand Load Instruction

- LD $r_5$, ($r_6$)
- Input register for generating input memory address
- Output register for generating Memory address

Two-Operand Store Instruction

- ST ($r_7$), $r_2$
- Input register
Encoding

- Assume that processor architecture is such that there are 16 registers.
- A two-operand machine instruction ADD r1, r2 will need 8 bits for operand.
- Opcode bits can be maximum for 256 possibilities, hence instruction size will be 16-bits and m = 2.
Encoding

• A three address machine three-operand instruction ADD r1, r2, r3 will need 12 bits for operand
• Opcode bits cannot be restricted to 4, because that will permit only 16 distinct processor instructions, hence in order to have many processor instruction with multiple addressing modes, the instruction size will become 24-bits and $m = 3$
Encoding

- \( I = \text{ADD}\ r_1 \) with AC (first source operand) implicit as accumulator AC register and \( r_0 \) (destination operand) also implicit as AC. \[\text{AC} \leftarrow \text{AC} + r_1\]

| Opcode 5-bits | \( r_1 \) 3-bits |
Immediate Operands and Offsets (Displacements)

- ST x(ri), rj, for example, ST –27(ri), rj

<table>
<thead>
<tr>
<th>Opcode 8-bits</th>
<th>ri 4-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>rj 4-bits</td>
<td></td>
</tr>
<tr>
<td>Displacement (Signed offset) 16 bit</td>
<td></td>
</tr>
</tbody>
</table>
Example a GPR-based program that performs the $2 + (7 \times 3)$

- First, we load the operands 2, 7 and 3 in the registers r1, r2, r3
  
  LD r1, (r0)+;
  LD r2, (r0)+;
  LD r3, (r0)+;
  MUL r2, r3 [#Result in r2];
  ADD r1, r2;
Encoding the combination of instructions
Combined instructions

- Conditions or no conditions for operation
- Two separate distinct operations of same type
Combined instructions

- Unsigned or signed integer specification of the operands
- Post operation modifications or no-modification of the processor status flags
- Source operand logical or arithmetic shifts before operation
- Immediate or displacement operands specification
Example: ARM Processor Multiply long, MUL and ADD long

- The destination has two registers, rdHi rdLo
- The source is register rs
- The other source is memory pointer register rm
- The instruction format— cond 00010 U A S rdHi rdLo rs 1001rm
Example: ARM Processor Multiply long, Mul and add long

- MULL operation is \( \text{rdHi} - \text{rdLo} \leftarrow \text{rm} \times \text{rs} \) and MLAL operation is \( \text{rdLo} - \text{rdLo} \leftarrow \text{rdLo} - \text{rdLo} + \text{rm} \times \text{rs} \)

<table>
<thead>
<tr>
<th>Cond 4-bits</th>
<th>Opcode 5-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>U, A, S 3-bits</td>
<td>Rd(Hi)Rd(Lo) 8-bits</td>
</tr>
<tr>
<td>Opcode 4-bits</td>
<td>Rs 4-bits Rm 4-bits</td>
</tr>
</tbody>
</table>
U, A and S bits

- U = 1 means signed, 0 means unsigned operation
- A = 1 specifies multiply and accumulate and 0 means multiply only
- S = 1 means that after the operation set the condition flags in the CPSR (current program status register) and = 0 means no change
Two types of Encoding of the instructions
Two Type of Encodings

Variable Length Instructions
- Usually CISCs

Fixed Length Instructions
- All Modern Processors and Usually RISCs

Machine Encodings
Fixed-length and Variable-length

- Examples of Fixed-length include stack-based architectures, because many operations do not need to specify their inputs, and Variable-length CISC architectures, which often contain a few instructions that can take a large number of inputs.
Variable-length instruction-set encoding

- Uses different numbers of bits to encode the instructions in the instruction set architecture, depending on the number of inputs to the instruction, the addressing modes used, and other factors
Instructions with variable length

- Multiple type of addressing modes and variation in the number of operands needed in a machine instruction
- There can be the decoding circuitry
- Automatically decodes the length required for the given operation for the opcodes
Advantage of variable length encoding

- Program length in memory is less when a limited number of operands are used in most instructions and a lesser number of instructions are used with immediate operands and offset bits.
Architect View

- Generally, architects want to find an encoding that is both compact and requires little logic decode, meaning that it is simple for the processor to figure out which instruction represented by a given bit pattern in the program.
Once the set of instructions that a processor will support has been selected, the computer architect must select an encoding for the instruction set architecture, which is the set of bits that will be used to represent the instructions in the memory of the computer.
Variable-length instruction-set encoding

- Each instruction takes only as much space in memory as it requires, although many systems require that all instruction encoding be an integer number of bytes long.
Variable-length instruction-set encoding

• Can reduce the amount of space taken up by a program, but it greatly increases the complexity of the logic required to decode instructions, since parts of the instruction, such as the input operands, may be stored in different bit positions in different instructions.
Variable-length instruction-set encoding

- Also, the hardware cannot predict the location of the next instruction until the current instruction has been decoded enough to know how long the current instruction is.
Fixed Length Encoding of instructions
Modern processors with 32-bit word length

- Memory not a constraint; therefore fixed length instructions preferred
- A processor e.g., the ARM processor, may provide for two instruction sets, one set with 32-bit instructions and another with 16-bit instructions (for example, Thumb® Instruction set)
- But there is a fixed length of instructions in each set
Fixed-length instruction set encoding

- Uses the same number of bits to encode each instruction in the instruction set architecture
Advantage of Fixed-length instruction set encoding

- Simple to decode
- Reducing the amount of decode logic required and the latency in the decode logic
Latency during execution

- Time period for which execution steps cannot proceed
- If decoding circuit uses a lesser number of gates in a simpler circuit, the latency also reduces during the execution
- Execution steps can proceed further only after decoding of the fetched instruction
Fixed-length instruction set architecture encoding

- Easily predict the location of the next instruction to be executed (assuming that the current instruction is not a branch)
- This makes it easier for the processor to use pipelining to improve performance by overlapping the execution of multiple instructions
Summary
We Learnt

- Fixed and variable length encoding
End of Lesson 17 on
Encoding of Instructions