• Register base-indexed addressing
  - Two address fields, A and R, both refer to the registers, which usually are the index register and the base register.
  - Base register: the only general-purpose register (GPR) which may be used for indirect addressing.
  - The actual address is the value in the base register + the value in the index register.

• Register base-scaled indexed addressing
  - Three address fields, two refer to the registers, one is the scale
  - The actual address is the value in the base register + (the value in the index register x scale).

Real Instruction Format

PDP-8
  - Released in 1965, discontinued in 1970
  - Simplest instruction design for general purpose computers
  - 12-bit fixed length, 12-bit words
  - A single GPR, Accumulator
  - Three instruction formats
  - Support 35 instructions
  - Adopt indirect, displacement, and indexed addressing
PDP-10
- Released in 1966, discontinued in 1983
- Designed for large-scale time-shared system
- Emphasis on making system easy to program regardless the hardware expenses
- Other elements of an instruction are independent of the opcode
- Each arithmetic data type should have a complete and identical set of operations
- Direct addressing
- 36-bit fixed length instruction
- 36-bit word length: 9 bits for opcode, 18-bit address field

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Register</th>
<th>I</th>
<th>Index register</th>
<th>Memory address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>9</td>
<td>12</td>
<td>14 17 18 35</td>
</tr>
</tbody>
</table>

I = indirect bit

PDP-11
- Released in 1970, discontinued in early 1990s
- Most popular minicomputer, the first officially named version of Unix ran on it
- Uses variable-length instructions
- 13 instruction formats, encompassing 0-, 1-, and 2-memory address instruction types
- Usually one word (16-bit) long. For multiple memory address instructions, 32- and 48-bit instructions are used
- 6 bits for register reference: 3 bits identify the register (employ 8 16-bit GPRs), and 3 bits for addressing mode
- Instruction set and addressing capability are complex. Increase hardware cost and programming complexity. But more compact program can be developed.
Wrapping Up

- We now know different types of operations and the ways to specify operands. The information is presented in an instruction format. The bits are grouped into fields, and the most common fields are: an operation code field, an address field, and a mode field.

- The number of address fields may be one, two, or three depending on the machine architecture and the type of ISA used.

- Based on the number of operand supported and the size of various fields, the length of the instructions vary.

- Some processors fit all the instructions into a single sized format, whereas others make use of formats of varying sizes.

- A machine belongs to a ISA class based on its internal storage space architecture. It can support a number of operations and can adopt several addressing modes.
Function Call Instructions
- Function (procedure) perhaps is the most fundamental language feature for abstraction and code reuse.
- Two basic instructions:
  - a call instruction: branches from the present location to the procedure
  - a return instruction: returns from the procedure to the place from which it was called

This diagram illustrates the execution sequence of function calls.

The execution sequence starts from the beginning of the main program. When the execution reaches at the instruction at address 4100, the instruction is CALL Proc1, invoke the procedure Proc1. Then, the execution jumps to the beginning of Proc1 and starts executing Proc1. Inside Proc1, the instruction at address 4600 is CALL Proc2. When the execution reaches at address 4600, it jumps to the beginning of Proc2 and starts executing Proc2. Once it’s done with the Proc2 and hits the RETURN instruction, the execution sequence goes back to Proc1 and keeps executing the remaining instructions in Proc1. In the remaining instructions, the instruction at address 4650 is CALL Proc2. So, the execution sequence jumps to the beginning of Proc2 and starts executing it again when it reaches at address 4650. The execution sequence goes back to where it left in Proc1 after it’s done with the execution of Proc2. When the execution sequence reaches at the last instruction of Proc1, which is RETURN, it’s done with the execution of Proc1. It then goes back to where it left in the main program and execute the remaining instructions in the main program.