Basic Assembly

Fetch-Execute Cycle

Increment the Program Counter

Fetch Instruction Addressed by Program Counter

Execute Fetched Instruction

Alter System State

Memory has a physicality

Memory Model

• `%rip` - Instruction Pointer (Program Counter)
• `%rbp` - Base Pointer
• `%rsp` - Stack Pointer
• Condition Codes:
  • `CF` - Carry Flag
    The most recent operation generated a carry out of the significant bit. Used to detect overflow for unsigned.
  • `ZF` - Zero Flag
    The most recent operation yielded a zero.
  • `SF` - Sign Flag
    The most recent operation yielded a negative value.
  • `OF` - Overflow Flag
    The most recent operation caused a two's-complement overflow—either positive or negative.

Assembly

• There are two types of operators in assembly.
  • **Assembly Instructions** — the instructions that make up the Instruction Set Architecture (ISA) and directly maps into executable code in memory.
  • **Assembler Directives** — special commands that instruct the assembler to perform additional work and serves to simplify assembly coding.
Basic Operation Types

- Data Movement/Storage: `movl`, `movq`, `pushq`, `popq`
- Data Manipulation: `addl`, `imull`, `sall`, `leaq`
- Control Flow: `cmp`, `test`, `jmp`, `jge`, `jl`

Assembler Directives

- These commands are unique to the assembler and typical start with a “dot”.
- A discussion of the directives can be found in the GNU Assembly Manual.
- Directives most common use is flagging specific information and performing common memory manipulations.

Instruction Size

<table>
<thead>
<tr>
<th>Intel Type</th>
<th>Suffix</th>
<th>Size</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>b</td>
<td>1</td>
<td>movb $0, %al</td>
</tr>
<tr>
<td>Word</td>
<td>w</td>
<td>2</td>
<td>movw $0, %ax</td>
</tr>
<tr>
<td>Double Word</td>
<td>l</td>
<td>4</td>
<td>movl $0, %eax</td>
</tr>
<tr>
<td>Quad Word</td>
<td>q</td>
<td>8</td>
<td>movq $0, %rax</td>
</tr>
</tbody>
</table>

Memory Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Form</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>$Imm</td>
<td>Imm</td>
<td>movl $0, %eax</td>
</tr>
<tr>
<td>Register</td>
<td>r</td>
<td>R[r]</td>
<td>movl %eax, %esi</td>
</tr>
<tr>
<td>Absolute</td>
<td>Imm</td>
<td>M[Imm]</td>
<td>movl $.LC1, %edi</td>
</tr>
<tr>
<td>Indirect</td>
<td>(r)</td>
<td>M[R[r]]</td>
<td>movl (%rax), %eax</td>
</tr>
<tr>
<td>Base+displace</td>
<td>Imm(r)</td>
<td>M[Imm + R[r]]</td>
<td>leaq -4(%rbp), %rax</td>
</tr>
<tr>
<td>Scaled Index</td>
<td>Imm(rb, ri, s)</td>
<td>M[Imm + R[rb] + R[ri] * s]</td>
<td></td>
</tr>
</tbody>
</table>
Jumps

<table>
<thead>
<tr>
<th>Name</th>
<th>Form</th>
<th>Condition</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>jmp label 1</td>
<td>1</td>
<td>Always jumps.</td>
</tr>
<tr>
<td>Indirect</td>
<td>jmp *label 1</td>
<td>1</td>
<td>Jumps to pointer.</td>
</tr>
<tr>
<td>Equal/Zero</td>
<td>je label ZF</td>
<td></td>
<td>Jumps on zero.</td>
</tr>
<tr>
<td>Greater/Equal</td>
<td>jge label ~(SF ^ OF) signed &gt;=</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less</td>
<td>jl label SF ^ OF signed &lt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **CF** - Carry Flag
  The most recent operation generated a carry out of the significant bit. Used to detect overflow for unsigned.

- **ZF** - Zero Flag
  The most recent operation yielded a zero.

- **SF** - Sign Flag
  The most recent operation yielded a negative value.

- **OF** - Overflow Flag
  The most recent operation caused a two's-complement overflow—either positive or negative.

---

Compile Process

- **Pre-Compiler**
- **Compiler**
- **Assembler**
- **Linker**
- **Source Code**
- **SourceCode w/ Expanded Macros**
- **Assembly**
- **Relocatable Obj Programs (multiple files)**
- **Executable**
- **Intermediate Form**
- **CC**
- **Fortran**
- **Java**
- **AMD Assembly Generator**
- **x86_64 Assembly Generator**
- **ARM Assembly Generator**
- **Front-End Compiler**
- **Back-End Compiler**
- **Pre-Compiler**
- **Source Code**
- **SourceCode w/ Expanded Macros**
- **Assembly**
- **Relocatable Obj Programs (multiple files)**
- **Executable**

---

GCC

- **Source Code**
- **Pre-Compiler**
- **SourceCode w/ Expanded Macros**
- **Compiler**
- **Assembly**
- **Relocatable Obj Programs (multiple files)**
- **Executable**