Basic Addition Algorithm

- Steps for addition (or subtraction):
  1. compute $Y_e - X_e$ (getting ready to align binary point). $Y_e > X_e$
  2. right shift $X_m$ that many positions to form $X_m \cdot 2^{e_e - e_m} + Y_m$
  3. compute $X_m \cdot 2^{e_e - e_m} + Y_m$

Example: $0.5372400 \times 10^2 - 0.1580000 \times 10^{-1} = 0.5370820 \times 10^2$

If result demands normalization, then normalization step follows:
  4. left shift result, decrement result exponent (e.g., 0.001xx...) continue until MSB of data is 1 (NOTE: Hidden bit in IEEE Standard)
  5. if result is 0 mantissa, may need to zero exponent by special step

Program Control Instructions

- Control over the flow of program execution and a capability of branching to different program segments
- One-address instruction:
  * Jump: direct addressing
  * Branch: relative addressing

<table>
<thead>
<tr>
<th>Name</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>BR</td>
</tr>
<tr>
<td>Jump</td>
<td>JMP</td>
</tr>
<tr>
<td>Skip next instruction</td>
<td>SKP</td>
</tr>
<tr>
<td>Call Procedure</td>
<td>CALL</td>
</tr>
<tr>
<td>Return from procedure</td>
<td>RET</td>
</tr>
<tr>
<td>Compare (by subtraction)</td>
<td>CMP</td>
</tr>
<tr>
<td>Test (by ANDing)</td>
<td>TEST</td>
</tr>
</tbody>
</table>

Example

- Adding operation on two IEEE single precision floating point numbers ($X$ and $Y$)
  \[ X = 01000000 \quad 101000000000000000000000 \]
  \[ Y = 11000000 \quad 011000000000000000000000 \]
  \[ S \quad E \quad M \]
  
  \[ N = (-1)^S \times 2^{E-127} (1.M) \]

\[ X = (-1)^0 2^{129-127} (1.01) = 2^2 \times 1.01 \]
\[ Y = (-1)^1 2^{128-127} (1.011) = -2^1 \times 1.011 \]
\[ X > Y \]
\[ Y = -2^{24}(1.011 \times 2^{-1}) = -2^{24}(0.1011) \]
\[ X + Y = 2^{24}(1.01 - 0.1011) = 2^{24}(0.1001) = 2^{1} \cdot 0.1001 \]
\[ = 01000000001000000000000000000000 \]

Conditional Branching Instructions

- May or may not cause a transfer of control, depending on the value of stored bits in the PSR (processor state register)

<table>
<thead>
<tr>
<th>Branch Condition</th>
<th>Mnemonics</th>
<th>Test condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch if zero</td>
<td>BZ</td>
<td>Z=1</td>
</tr>
<tr>
<td>Branch if not zero</td>
<td>BNZ</td>
<td>Z=0</td>
</tr>
<tr>
<td>Branch if carry</td>
<td>BC</td>
<td>C=1</td>
</tr>
<tr>
<td>Branch if not carry</td>
<td>BNC</td>
<td>C=0</td>
</tr>
<tr>
<td>Branch if minus</td>
<td>BN</td>
<td>N=1</td>
</tr>
<tr>
<td>Branch if plus</td>
<td>BNN</td>
<td>N=0</td>
</tr>
<tr>
<td>Branch if overflow</td>
<td>BV</td>
<td>V=1</td>
</tr>
<tr>
<td>Branch if no overflow</td>
<td>BNV</td>
<td>V=0</td>
</tr>
</tbody>
</table>
### Conditional Branching Instructions (Contd.)

- **Unsigned or signed numbers**

<table>
<thead>
<tr>
<th>Branch Condition</th>
<th>Mnemonics</th>
<th>Condition</th>
<th>Status bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch if higher</td>
<td>BH</td>
<td>A &gt; B</td>
<td>C + Z = 0</td>
</tr>
<tr>
<td>Branch if higher or eq</td>
<td>BHE</td>
<td>A ≥ B</td>
<td>C = 0</td>
</tr>
<tr>
<td>Branch if lower</td>
<td>BL</td>
<td>A &lt; B</td>
<td>C = 1</td>
</tr>
<tr>
<td>Branch if lower or eq</td>
<td>BLE</td>
<td>A ≤ B</td>
<td>C + Z = 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Branch Condition</th>
<th>Mnemonics</th>
<th>Condition</th>
<th>Status bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch if greater</td>
<td>BG</td>
<td>A &gt; B</td>
<td>N, V, Z = 0</td>
</tr>
<tr>
<td>Branch if greater or eq</td>
<td>BGE</td>
<td>A ≥ B</td>
<td>N, V = 0</td>
</tr>
<tr>
<td>Branch if less</td>
<td>BL</td>
<td>A &lt; B</td>
<td>N, V = 1</td>
</tr>
<tr>
<td>Branch if less or eq</td>
<td>BLE</td>
<td>A ≤ B</td>
<td>N, V + Z = 1</td>
</tr>
</tbody>
</table>

---

### Procedure Call and Return Instructions

- **Procedure:** self-contained sequence of instructions that performs a given computational task
- **Call procedure instruction:** one-address field
  - Stores the value of the PC (return address) in a temporary location
  - The address in the call procedure instruction is loaded into the PC
- **Final instruction in every procedure:** return instruction
  - Take the return address and load into the PC
- **Temporary Location:** fixed memory location, processor register or memory stack
  - E.g. stack
    - Procedure call: SP ← SP - 1; M[SP] ← PC + 4; PC ← Effective address
    - Return: PC ← M[SP]; SP ← SP + 1

---

### Program Interrupt

- **Handle a variety of situations that require a departure from the normal program sequence to another service program, similar to a call procedure**
- **Different from procedure calls:**
  - Initiated at an unpredictable point in the program, rather than the execution of an instruction
  - Address of the interrupt service is determined by a hardware procedure
  - The information that defines all or part of the contents of the register set, rather than only the PC, should be stored temporarily
- **After finishing interruption, resume to the same state before the interruption**
  - PSR: other than condition codes, also contains what interrupts allowed, user/system mode indication, etc.

---

### Type of Interrupts

- **Hardware interrupts**
  - **External interrupts:**
    - Input/output devices requesting transfer of data
    - Timing devices time-out event
    - Circuit monitoring the power supply detect an impending power failure, in the ISP transfers the register set contents to nondestructive storage like disk, etc.
    - Any other external source
  - **Internal interrupts (traps):**
    - Invalid or erroneous use of an instruction
    - Arithmetic overflow, attempt to divide by zero, an invalid opcode, memory stack overflow, protection violation
- **Software interrupts:** initiated by executing an instruction
  - System call instructions, change from user mode to system mode
If EI=1, and current instruction is completed, acknowledge interrupts by:

- \( \text{SP} \leftarrow \text{SP} - 1 \)
- \( \text{M}[\text{SP}] \leftarrow \text{PC} \)
- \( \text{SP} \leftarrow \text{SP} - 1 \)
- \( \text{M}[\text{SP}] \leftarrow \text{PSR} \)
- \( \text{EI} \leftarrow 0 \)
- \( \text{INTACK} \leftarrow 1 \)
- \( \text{PC} \leftarrow \text{IVAD} \)

External Interrupts

中断处理

中断向量地址

中断确认

中断处理单元 (CPU)

结束指令执行

使能中断

中断确认寄存器

程序计数器

内存栈