Basic Assembly Language

Operations – actions the computer carries out

Operands – data acted on
Operands – the data acted on by instructions

• Where are they?

Processor has internal locations for storing data or addresses: registers
Each register holds a standard unit: double word = 64 bits
Limited number of registers: 32 for ARMv8

Memory is separate from processor
Has almost unlimited space for programs and data
Addressed like a giant array, each byte is addressed
Instructions – a load/store architecture

• Instructions for arithmetic and logical operations only work on values in registers or given as part of the instruction (immediate)
  • Simpler instructions
  • Faster access to data
  • For ARMv8 designated X0, X1, X2, ... X31

• Separate instructions move data from/to memory locations
  • Load bring data from a memory location to a register
  • Store sends data to a memory location from a register
Arithmetic instructions

• Take three parameters:
  • Destination for result (always a register)
  • Two operands
    • First is always a register
    • Second may be a register or immediate

Example: add X11, X9, X10
Takes the values found in X9 and X10, sums them and puts the result into X11

Example sub X19, X10, #8
Takes the value in X10, subtracts the value 8, places result into X19
More complex expressions take more instructions to complete

Example:  \[ g = a + b + c + d \]

Assume a is in X9, b in X10, c in X11, d in X12, g in X19

\[
\begin{align*}
\text{add X19, X9, X10} & \quad \text{// X19 = a + b} \\
\text{add X19, X19, X11} & \quad \text{// X19 = (a+b) + c} \\
\text{add X19, X19, X12} & \quad \text{// X19 = ((a+b) + c) + d}
\end{align*}
\]

Note: “//” indicates the rest of the line is a comment ignored by system
Can you do this?

Example:  $g = (a - b) + (c - d)$
Assume $a$ is in X9, $b$ in X10, $c$ in X11, $d$ in X12, $g$ in X19
Answer

Example: \( g = (a - b) + (c - d) \)

Assume a is in X9, b in X10, c in X11, d in X12, g in X19

\[
\begin{align*}
\text{sub } X13, X9, X10 & \quad // X13 = a - b \\
\text{sub } X14, X11, X12 & \quad // X14 = c - d \\
\text{add } X19, X13, X14 & \quad // X19 = (a - b) + (c - d)
\end{align*}
\]
What if information is in memory

Suppose we have an array, A[ ], in memory

<table>
<thead>
<tr>
<th>Address</th>
<th>value</th>
<th>&lt; A[i] &gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>10008</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>10016</td>
<td>-12</td>
<td></td>
</tr>
</tbody>
</table>

10000 is the “base address” of the array

Since each data item is 64 bits and memory is addressed by bytes (8 bits), each item in the array has the address increased by 8
How do we do this?

Accessing the array $A$

First assume that we have 1000 placed in register $X20$ (the base address).

Then

$\text{ldur } X9, [X20, \#0] \; // \text{ puts the first array value into register } X9$

$\text{ldur } X10, [X20, \#8] \; // \text{ puts the second array value into register } X10$

$\text{add } X11, X9, X10 \; // \text{ would add the two array values}$

$\text{stur } X11, [X20, \#16] \; // \text{ would store the result into the memory for the }$

$\text{ } \; // \text{ third array element (overwriting any value there before)}$

To operate on array elements, they first must be loaded ($\text{ldur}$). Results that go into the array must be stored after they are calculated ($\text{stur}$).