Review Problem 3

- In assembly, compute the average of positive values X0, X1, X2, X3, and put into X10

```
ADD  x10, x0, x1
ADD  x11, x2, x3
ADD  x10, x10, x11
LSR  x10, x10, #2
ADDI x12, x31, #4
SDIU x10, x10, x12
ADDI x12, x31, #4
// Goal, but there is no SDIVI
```
Memory Organization

Viewed as a large, single-dimension array, with an address. A memory address is an index into the array. "Byte addressing" means that the index points to a byte of memory.
Memory Organization (cont.)

Bytes are nice, but most data items use larger units.
  Double-word = 64 bits = 8 bytes
  Word = 32 bits = 4 bytes

<table>
<thead>
<tr>
<th>0</th>
<th>64 bits of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>64 bits of data</td>
</tr>
<tr>
<td>16</td>
<td>64 bits of data</td>
</tr>
<tr>
<td>24</td>
<td>64 bits of data</td>
</tr>
</tbody>
</table>

Registers hold 64 bits of data

$2^{64}$ bytes with byte addresses from 0 to $2^{64}-1$

$2^{61}$ double-words with byte addresses 0, 8, 16, ... $2^{64}$-8

Double-words and words are aligned
  i.e., what are the least 3 significant bits of a double-word address?

---

---
Addressing Objects: Endian and Alignment


<table>
<thead>
<tr>
<th></th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>$2^{63}, 2^{56}$</td>
<td>$2^{55}, 2^{48}$</td>
<td>$2^{47}, 2^{40}$</td>
<td>$2^{39}, 2^{32}$</td>
<td>$2^{31}, 2^{24}$</td>
<td>$2^{23}, 2^{16}$</td>
<td>$2^{15}, 2^{8}$</td>
<td>$2^{7}, 2^{0}$</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Big Endian: address of most significant byte = doubleword address
Motorola 68k, MIPS, IBM 360/370, Xilinx Microblaze, Sparc

Little Endian: address of least significant byte = doubleword address
Intel x86, DEC Vax, Altera Nios II, Z80

ARM: can do either – this class assumes Little-Endian.
Data Storage

Characters: 8 bits (byte)
Integers: 64 bits (D-word)
Array: Sequence of locations
Pointer: Address (64 bits)

// G = ASCII 71
char a = 'G';
int x = 258; \( 258 = 2 + 1 \times 256 \)
char *b;
int *y;
b = new char[4];
y = new int[10];

(Note: real compilers place local variables (the “stack”) from beginning of memory, new’ed structures (the “heap”) from end. We ignore that here for simplicity)
Loads & Stores

Loads & Stores move data between memory and registers

All operations on registers, but too small to hold all data

LDUR X0, [X1, #14]  // X0 = Memory[X1+14]
130 + 14 = 144

STUR X2, [X3, #20]  // Memory[X3+20] = X2
4 + 20 = 24

<table>
<thead>
<tr>
<th>General Purpose Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0: 66</td>
</tr>
<tr>
<td>X1: 130</td>
</tr>
<tr>
<td>X2: 723</td>
</tr>
<tr>
<td>X3: 4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>24: 723</td>
</tr>
<tr>
<td>144: 66</td>
</tr>
</tbody>
</table>

Note: LDURB & STURB load & store bytes
Addressing Example

The address of the start of a character array is stored in X0. Write assembly to load the following characters

\[ X2 = \text{Array}[0] \]

\[ \text{LDURB } x2, [x0, #0] \]

\[ X3 = \text{Array}[1] \]

\[ \text{LDURB } x3, [x0, #1] \]

\[ X4 = \text{Array}[2] \]

\[ \text{LDURB } x4, [x0, #2] \]

\[ X5 = \text{Array}[k] \quad // \text{Assume the value of } k \text{ is in } X1 \]

\[ \text{ADD } x5, x0, x1 \quad // x5 = & \text{array}[k] \]

\[ \text{LDURB } x5, [x5, #0] \]