|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Public class Bit extends Rock | Public class CPU extends Actor | Public abstract class Bus extends Bug |  |  |
| Public void flip()  Public boolean get()  Public void set(boolean val) | Public CPU()  Public void act() | Public void act()  Public void move()  Public abstract void processBit(Bit b, int p) |  |  |
| Private boolean val  Public static final Bit ZERO  Public static final Bit ONE | Private Bus bus | Private short instruction  Private byte dat |  |  |

I will be building a computer in GridWorld. The CPU will be represented by an Actor. The CPU has a small amount of memory in it. This memory is divided into registers. The CPU understands 32-bit (int) instructions. Each int value represents a different instruction for the CPU to execute. Instructions are stored alongside other memory. This is known as a stored-program computer. The address of the next instruction is stored in the 12-bit program counter, which is one of the registers. Every time the act method is called, the CPU fetches the next instruction from memory and then increments the program counter. The fetched instruction is then executed.

The Bus is a sort of worker that communicates between CPU and memory. It has a short containing the 12-bit memory address. It also has a byte to hold data that is read or to be written. Subclasses of Bus exist to provide different functions, such as read from memory, write to memory, arithmetic operations, bitwise operations, etc.

Memory will be displayed as rows of Bits on an UnboundedGrid. I would like to implement 12-bit memory addresses, so the number of unique memory addresses is 2^12 = 4096. If each memory address indicates a byte of memory then the maximum memory capacity for this processor is 4096 bytes = 4 kB. Memory allocation will be handled using the “buddy system.” A description of this system can be found at <http://en.wikipedia.org/wiki/Buddy_memory_allocation>.

This project can be used to visually show what a CPU is, what memory is, and how the two interact. It would show how the CPU reads from, writes to, allocates, and frees memory among other tasks. This project can also help students understand efficiency of code. When programming in a high level language like Java, the underlying machine instructions are often concealed. Tasks that are one line of code in Java may actually be several instructions for the machine. The Bus Actors help to show this. Each Bus represents a task executed by the CPU. So when a student tells the CPU actor to do a task in Java, the student sees how the task breaks down on the machine level and understands the efficiency of different operations.