# Minutes

24 March 2015

* Functions
  + How are functions represented in bytecode?
    - This needs to be formalized
  + Function calls, how?
    - This also
  + Review function instructions and their purpose
    - These will change once data representation has been formalized
* Queries about instruction definitions
  + Review memory access instructions and their purpose
    - These will change once data representation has been formalized
  + 16b ptr values. After instruction there is a 16bit relative pointer. Is it relative to the beginning of the instruction, the end of the instruction, or the end of the pointer? Especially relevant for jmp which has three. If relative to the beginning of the number, then non-jumps have weird addresses like 2 for the last operand, 4 for the second last, etc. With this reasoning in mind, we should do it rel to the end of the instruction
    - Pointers are relative to the end of the instruction with which they are associated
  + null pointer cab: what is null here?
    - That’s just what an assembler might see in assembly, which it will interpret as this instruction. My renaming defeats the point of having null as an operand because the instruction has null in the name
  + cshl has a 16 bit immediate value?
    - That should be a 64 bit immediate
  + We should add a halt instruction
    - We already do, except it’s called ret
  + Can we just have separate register mapping for pointers and integers? We have the space  
    ie have totally distinct ri and pi registers
    - We could, but then we sacrifice applicability to other architectures. Something to talk about in the next RT
* Feedback
  + There are two types of prediction involved in branches right? The first is branch prediction which is prediction of whether a conditional branch will be taken or not. The second is branch target prediction, which is where the branch will go. Branch target prediction is not needed for direct branches, conditional or not, because that’s in the instruction so there’s no need to guess
    - Yep you got it
* Things to get done:
  + Two control virtual machines can be written. The one has the same dispatch process, but no register mapping. This can be used to test how significant the unique dispatch we have is. The other virtual machine will be totally ordinary and will test overall usefulness.
  + Think about how to test instructions for correctness. Some options:
    - Write a test framework in C. This framework will interface to the virtual machine though a C-conforming function in assembly that will perform one instruction and return the VM state instead of dispatching. The C framework will evaluate the change in VM state to determine whether the instruction executed correctly
    - Someone else can write the same code, and results can be compared.
    - Some sort of test program that can utilize all instructions.