We have 4 weeks until the deadline so we should talk about how to split the work between us. I’m happy for us to do everything sort of together and split the work as we go along depending on who is more comfortable doing what, or we could also make a more defined plan early on.

Main tasks:

1. Initial stages
   * Deciding what the instructions will be and making a general block diagram for the hardware
   * Agreeing on addressing modes to enable linked list traversing
2. Implementation
   * Software: writing “benchmark algorithms” in assembly.
   * Hardware: implementing multiplication block for linear congruential generator (kind of hard)
   * Hardware: implementing stack to enable Fibonacci algorithm
   * Pipelining?
3. Final stages (each worth 50% of project)
   * Writing report
   * Recording video demonstration

Clarke suggests:

* Splitting the team so one person is responsible for software and two responsible for hardware (implementing the ISA on Quartus).
* One of us acts as main editor of the report (but software/hardware sections can be written by individual people working on them)

As far as I can see there’s two main things we need to focus on to figure out the design and these are some ways I’ve thought about maybe implementing them:

1. Implementing a stack so that we can use recursion in the software
   * <https://www.geeksforgeeks.org/subroutine-subroutine-nesting-and-stack-memory/>
   * Stack register containing pointer to “top” of stack
   * Section of memory dedicated to contain the stack
   * Two options: PUSH/POP instructions or subroutine call instructions that jump to and return from subroutines (do PUSH/POP automatically). Relevant for pipelining
2. Figuring out a way to implement multiplication in hardware
   * Basic implementation has too much propagation delay (proportional to word size)
   * Booth’s algorithm
   * Wallace trees
   * Dadda multipliers
   * Area/power/speed trade-offs, content for report, referencing papers: <http://www.iosrjournals.org/iosr-jece/papers/vol1-issue1/H0114348.pdf?id=4582>

Other things to discuss:

1. Pipelining
   * AVR architecture runs every instruction in two cycles so there are no breaks in the pipeline