- Identifies you and your programming partner by name

Dylan Chelo and Nick Powers completed this assignment.

- Acknowledges help you may have received from or collaborative work you may have undertaken with others

We received an extension on the assignment till Monday. On top of this we used the rumload module that was given to us as well as some more code from the rumdump lab. However we did not participate in any collaborative work with anyone but each other.

- Identifies what has been correctly implemented and what has not

We believe we were able to fully implement the entire universal machine.

- Briefly enumerates any significant departures from your design

1. We didn't end up copying the fields from the rumdump lab
2. We ended up implementing a custom struct with some functions called segment manager to hold the hashmap of segments of memory as well as the unmapped segments in a vec.
3. We also decided against using a function for load value because of the unique word format. It just made more sense to have an if statement to catch that at the beginning since we did not use the field struct previously mentioned from the rumdump lab.

- Succinctly describes the architecture of your system. Identify the modules used, what abstractions they implement, what secrets they know, and how they relate to one another. Avoid narrative descriptions of the behavior of particular modules.

We used two modules for our universal machine, one being the rumload module which handles input and the other being the main module holding the universal machine and everything else. Looking at the rumload module, it does not hold any secrets as its one function it uses is public.

It implements the following abstractions: reading the binary file of u32 words, reading the data from the file and storing it, converting data into u32's, and putting the u32's in a vector. As for the main module it relates to the rumload module by calling its function to read the binary file. Since this module is the main module I don't know if I would consider there to be any secrets because usually this is the one place where there are not any secrets, but if you did not have access to the source code the only public functions are the ones for the segment manager struct. As for abstractions they are here: the segment manager is a struct used to store, allocate, and deallocate memory, the segment struct represents a segment of memory, and the opcode enum represents the numerical value for each assembly instruction. There are also abstractions implemented for the different opcodes but that does not have much to do with the actual architecture of the code. As for the architecture of the project as a whole, the input is gathered then a function that loops through the instructions is called and it calls the appropriate function for each opcode until there are no more.

- Explains how long it takes your UM to execute 50 million instructions, and how you know

Our UM will take approximately 68.65 seconds to execute 50 million instructions, we know this because it took 117.16 seconds to run midmark.um which has 85,070,522 instructions. Given these data points we just found out that it takes approximately 0.000001373 seconds per instruction and multiplied that by 50 million, which gives us 68.65 seconds.

- Says approximately how many hours you have spent analyzing the assignment

We spent approximately 2 hours analyzing the assignment before planning the design.

- Says approximately how many hours you have spent preparing your design

We spent approximately 4 hours preparing our design for this assignment.

- Says approximately how many hours you have spent solving the problems after your analysis

After analyzing the problem it took us approximately 2 hours to prepare solutions on pen and paper and 16 hours to actually code everything.