Michael Palumbo

October 28th, 2018

CS 164 Section 004, Brain L. Stuart

A Look at The Apollo Guidance Computer

The creation of the Apollo Guidance Computer not only allowed humanity to set foot on the moon, but also engaged and improved the importance of software engineering and software development strategies. The National Aeronautics and Space Administration (NASA) teamed up with Massachusetts Institution of Technology to create the technology and software that was used in the AGC. This feat was not easy due to limited memory and mistakes made during the development period. But also because of that they developed some very clever ways of utilizing bits and user interactions.

The Apollo Guidance Computer, or AGC, was the central computer that was installed onto the Command Module. The Command Module was the main canopy that the astronauts sat in to control and the ship. If the space ship were to be landing on the moon, the ship would also have another AGC located in its lunar module, as well as an Abort Guidance System (AGS) which could get the ship off the moon if the AGC in the lunar module failed. The AGC stored most of its program code in read-only format, but a small amount of code did reside in RAM which meant you could edit it during the trip. About 69,120 bytes of data was story in memory, and only 3,840 bytes were stored in ram. “Words” are groups of data, the AGC defined a word as fifteen bits of data and one bit as a parity bit (the parity bit ensure that a string of bits is either even or odd). This meant there were 2048 words of RAM, and 36,864 words of read-only memory.

If you were to look at a “word” you would see a series of sixteen bits, the left most bit is fifteen and we count down the bits left to right, after one we will have one bit left, the last bit will be the previously mentioned parity bit. Arithmetic operations performed by the CPU follows a Single-Precision Format, which is very similar to One’s-Complement Notation. Like One’s-Complement, the left most bit in Single-Precision Format determines whether the word is negative or positive, leaving bits fourteen to one as the represented data. If the fifteenth bit is one, then the magnitude of the word is negative and vise versa. Like binary or decimal, the left most bit has the highest value, but unlike them, all the bits are represented in a fraction. Where the fourteenth bit represents a half, the thirteen represents a fourth and so on to the power of two. But being this is a space shift, we may want to be more precise; this is where the Double-Precision Format comes into play. Double-Precision format allows us to combine two words to get a total of twenty-eight bits to represent the magnitude. The fourteen bits in word one will represent more-significant bits, meaning the second word bits would be a continuation of the powered fractions, allowing us to get more precise factors. The fifteenth bit in word one will determine the overall magnitude of the word, if the fifteen bit of the second word matches the first word, the two words are added together, if they differ, they subtract. But to even play with these values (or at least the values located in RAM) we’ll have to look at how to interact with the computer.

The Apollo Guidance Computer uses a display and keyboard, called DSKY, as an interface. But this wasn’t a standard monitor and keyboard we are used to; the DSKY monitor has two panels, the left pane are the warnings lights, like if the computer could not find an altitude reference or there was an operation error; and the right panel shows more specific information but all in numbers, like what is the altitude is. The keyboard is mostly an array of numbered keys, a verb and noun key, and a few others. To input commands into the computer you would have to specify a verb (an action to do) and a noun (mem location that it’s acting upon). The commands are all in numbers and astronauts would have to memorize them. Verbs and nouns are two-digit numbers that the astronaut would insert by clicking the verb key, entering the two number, if it was six they would type zero six, and then press the noun key and enter two numbers and then press enter. Depending on the numbers they pressed certain action would commence. If it was verb zero-six, noun thirty-six, it would display the AGC’s internal clock; because zero-six means to show the information of the noun on the display and thirty-six is where the AGC clock is saved. Some verbs would not need an accompanying verb such as seventy-five which will initiate launch. With this assortment of verbs and nouns astronauts can find out anything about their ship’s trajectory and do calculations, like how much to tilt the ship.

The AGC was a priority-interrupt system and was capable at handling multiple jobs at once. The AGC will always execute the job with the highest priority first. There are 2 types of job scheduling in the AGC: Executive and Waitlist. The Waitlist Schedule can only handle nine short *tasks*, not jobs. Waitlist dedicates four milliseconds to execute a task, if it takes any longer the Waitlist will promote the task into a job and will move it to the Executive queue. The Executive Schedule manages the DSKY display and can handle up to seven jobs at once and will search every 20 milliseconds for a job or task with higher priorities than the current ones. If the Executive can find no more jobs to run, it’ll execute a program called “Dummy Job” continuously until another job comes into the queue, the dummy job is like a sleeping protocol. Once jobs are finished executing they are placed into twelve possible erasable memory locations. Software made for the AGC would spark the realization of how important software developers are in the field of computing and would even coin the name “software engineer”. These jobs were very clever in minimizing space but in doing so it caused confusion and conflict during the software development which later cause even more problems.

NASA and MIT had never worked on a software development project to this scale, NASA also pushed MIT because they didn’t trust that MIT could deliver reliable software on time. This push made code poor and bugs plentiful, which cause delays to fix them. The design orientation wasn’t thought out enough and programmers made alterations to existing code to fix their problems, which meant other people’s programs wouldn’t work with the changed source code. Another problem was that the memory was very limited, to over come this MIT created complex and obscure code which made it difficult to understand the programs and validating them, which caused more delays. The delays made NASA worried that they were falling behind on schedule, so they decided to hire more programmers. This meant that more money had to be spent hiring them and more time spent communicating between all the programmers, so not good at all. Looking back on this, scholars learned a lot of important lessons from NASA’s mistakes, such as requirements must be clearly defined and carefully managed and more programmers does not equal faster development.

In the end, everything worked out fine, it got a people to the moon and back. Today, the Apollo Guidance Computer is a marvel to look at for it’s clever use of bits and was even revolutionary in computing history, being one of the first computers able to do multiple jobs at once. The Apollo Guidance computer wasn’t just revolutionized technology but also how we approach the development process. The Apollo Guidance Computer will always remain as a stable of human ingenuity.

Sources:

“Computer in Spaceflight: The NASA Experience.” NASA, [www.history.nasa.gov/computers/Ch2-6.html](http://www.history.nasa.gov/computers/Ch2-6.html).

Burkey, Ron. Virtual AGC Assembly-Language Manual, Ibiblio, [www.ibiblio.org/apollo/assembly\_language\_manual.html](http://www.ibiblio.org/apollo/assembly_language_manual.html).

Burkey, Ron. Virtual AGC Assembly-Language Manual, Ibiblio, [www.ibiblio.org/apollo/assembly\_language\_manual.html](http://www.ibiblio.org/apollo/assembly_language_manual.html).