SE3910 – REAL TIME SYSTEMS

Memory Requirements and

July 20, 1969: The day software bugs nearly lost us the moon

ROADMAP

- Monday
  - Memory Utilization
- Wednesday
  - Toyota systems failure
- Friday
  - Exam review and course wrapup

OBJECTIVES

- Explain the 1201 and 1021 alarms encountered on Apollo 11 and explain the relevance to real time systems.
- Explain how to calculate total memory utilization
- Explain how to limit memory utilization

- What types of memory usage do we have on our system?

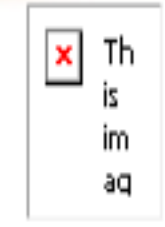
MEMORY UTILIZATION

$$M_T = M_{PG} \cdot P_{PG} + M_{ST} \cdot P_{ST} + M_{DT} \cdot P_{DT} + M_{PM} \cdot P_{PM},$$

MEMORY UTILIZATION

MEMORY ALLOCATION

$$M_A = \frac{U_A}{T_A}, \quad A \in \{T, PG, ST, DT, PM\},$$



TOTAL MEMORY UTILIZATION

Example: Total Memory Utilization

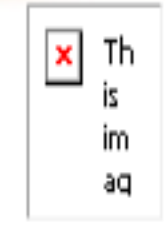
Suppose, a soft real-time system has 64 M bytes of program memory that is loaded at 75%, 16 M bytes of data memory that is loaded at 25%, and 8 M bytes of stack area that is loaded at 50%. All these memory-loading figures represent the corresponding worst-case values. Besides, there is no separate parameters area in this particular memory configuration. Thus, the total memory utilization can be calculated by Equation 7.23

LIMITING MEMORY UTILIZATION

- Avoid recursion
  - Uses up a lot of stack space
- Avoid memory fragmentation
  - Avoid allocating and deallocating memory unnecessarily
- Carefully manage the scope of variables
  - Helps to control stack utilization
- Optimize memory usage with registers
  - Compiler setup and options
- Estimate your memory usage before starting a project
  - Helps to gauge are you using things efficiently

MEMORY USAGE

- Lets assume I have the following data
  - Person's name
  - Person's DOB
  - Person's Address
  - Person's Phone #
  - Person's user ID
  - Person's Password (Stored in hashed format)
  - A set of photos
    - Photo
    - Description
    - Date
    - Time
    - Users tagged in the photo



VIDEO

APOLLO 11 LANDING
TRANSCRIPT

And, of course, it very nearly didn't

- **Computer alarms on descent; threat to landing abort**
- **Manual takeover at 1300 ft (90 secs of fuel)**
- **4 miles downrange, bolder field**
- **Heart pounding (156 beats per minute) Armstrong landed with only seconds of fuel to spare.**

102:38:26 Armstrong: Program Alarm. (6k ft agl)
102:38:30 Armstrong: It's a *1202*.
102:38:32 Aldrin: *1202*.
102:38:42 Armstrong (To Buzz) What is it? Let's incorporate (the landing radar data). (To Houston) Give us a reading on the *1202* Program Alarm.
102:38:53 Duke: Roger. We got you...(With urgency) We're Go on that alarm.
102:39:14 Aldrin: Same alarm.. appears to come up when we have a 16/68 up.
..
102:42:08 Duke: Roger. Copy.. Eagle, Houston. You're Go for landing. Over
..
102:42:17 Aldrin: Roger. Understand. Go for landing. .. Program Alarm
..
102:42:22 Aldrin: *1201* (3k ft)
102:42:24 Armstrong: *1201*!
102:42:25 Duke: Roger. *1201* alarm. (Pause) We're Go. Same type. We're Go.
..
102:45:58 Armstrong Houston, Tranquility Base here. The Eagle has landed.

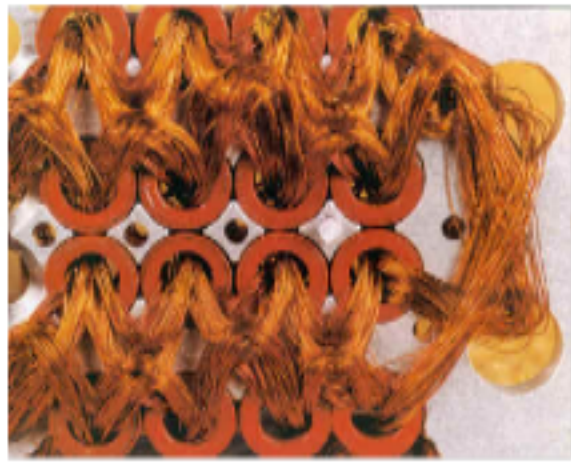
FROM *GENE KRANZ:* "FAILURE IS NOT AN OPTION"

- Final simulation done prior to the launch,
  - Dave Scott and Jim Irwin in the LM simulator.
  - landing simulation was aborted - unnecessarily
  - because of a 1201 program alarm
- Kranz sent Bales off to work up rules for each type of alarm. Later that evening, Bales rings Kranz saying
  - "We should not have aborted (due to that guidance system error)"

Th is im aq

INSIDE THE APOLLO COMPUTERS

- **LEM/CM** computer's had two types of memory:
  - fixed memory
    - programs, constants and landmarks
    - 36,864*15 bit words= 74KB (!!)
  - erasable memory,
    - variables/ registers used in calculations
    - 2,048 15-bit terms.
    - coincident-current ferrite cores woven into a rope with copper wires and sealed in plastic.
- Real-time multi-tasking operating system.
  - Always processes the job with the highest priority before other, lower priority jobs



- Two Apollo control programs :
  - Waitlist handled <= 9 quick tasks (4ms or less)
  - Executive handled longer tasks (up to 7 tasks)
- Each tasks had erasable memory
  - Memory was shared (up to seven ways!)
  - 1202 error: indicating a CPU overload,
- On descent: searching for rendezvous radar data
- Not fatal
  - Computer had been programmed to recognize this task as being of secondary importance
  - Ignored it, performed other tasks instead

WHAT WERE THE 1202/1201 ALARMS?

- M.I.T. Instrumentation Laboratory ("the Lab")
  - Built the Apollo Guidance and Navigation System;
- 10 seconds after “the eagle has landed”,
  - NASA rang the lab
  - "What were those alarms?”
  - “We're launching in 24 hours and we're not going with alarms.”
  - “We must have an operational computer!”

MASSIVE DEBUGGING AT M.I.T.

- The bug: “cycle stealing”
- Overload of queue
  - computer not getting to certain computations,
- What was slowing things up?
  - I/O system keeps looking for data.
  - The Rendezvous Radar Switch was in the AUTO position and the computer was doing I/O looking for radar data.
- Error in the crew procedures
  - “Place rendezvous radar switch” to “AUTO” during descent WRONG!
- Why not seen found during simulation?
  - The switch was not connected to a real computer (procedures validation performed on functional simulation)
- Last message before lunar take-off
  - Glenn Lunney,(Flight Controller), calmly told the astronauts…
  - "Please put the Rendezvous Radar Switch in the Manual position".

Th is im aq

LESSONS FOR SQA

- Bad software can kill good hardware
- Manuals matter
- Test what you fly (and nothing else)
- Do your criticality analysis right
  - For descent, rendezvous radar was apparently not-critical
    - Rendezvous radar used post ascent, not descent
    - Also, even if it failed on ascent, then just launch to lunar orbit and let CM's systems do the docking.

ONE FOOTPRINT IS LONELY…