Stephen Wilson

Usability Study.

1: Imaginary Users

Wephen Stilson is an aspiring computer science student who loves the subject. He particularly enjoys seeing how things work as they happen. A supplement to this trait is that he learns much better from a visual and hands on example rather than from a lecture with a few power points. By using this software, Wephen will be given a visual and hands on way of learning how ARM assembly works through memory as programs are being executed. The major feature that he will benefit from will be the interface that shows what is being stored in each register.

Jondrew Masterdaughter is a student who likes to be in control of every aspect of a software that he uses. He discovered that he will be learning ARM assembly in his 305 course. The one thing he dreads is that the execution of the programs he writes will leave him in the dust. The feature he will benefit from in this software will be the “STEP” function that allows a step-by-step execution of the ARM program that will be loaded into the emulator software.

Anna Kondha is a rather forgetful student who does not like repeating her progress. She maintains an expanding folder that contains all of her written notes and assignments. When hearing about the ARM assembler being used for CS 305, she immediately becomes concerned that something might go wrong during the program’s execution (whether it was in the emulator itself, or an outside distraction). She wants to be able to save her progress of where the execution instructions were last held. She will obviously benefit greatly from the program’s save feature.

2: Overview Paragraph

ARRM(working title) is a software that emulates (behaves like) the ARM processor. ARRM is being developed for a college professor for the purpose of teaching a systems architecture class. The ARRM software is meant to be used by computer science students and/or hobbyists who would love to see exactly how the ARM assembly instructions are handled during execution. The user will load an ARM assembly program into the ARRM emulator. Once the program begins operation, the software’s user interface will display each memory register of an ARM processor, and will show the user what instruction is in that register at that moment in time. Additionally, ARRM will allow the user to control the execution of the program one “step” at a time. This means that ARRM will store and execute one instruction and pause its progress until the user presses a big button. This “step” button will prompt the next action in the

instruction execution. Another major feature of ARRM is the ability to save the progress of execution and load it up at a later point in time. In all, this hands-on approach to teaching and learning ARM assembly employs the concept of “learning at one’s own pace”, and will prove to be truly beneficial to those who require a visual and interactive learning experience.

3: Paragraph Response: Focus Group 1

Because of the nature of this project, I decided it would be easier for my non CS focus group members to have a small context provided. (What is arm, how it will be used) I wanted them to focus on using the application rather than knowing anything about computers. In this case, four of my focus group volunteers know nothing about the material, but found it easy to think about my questions after a context was given.

1: Frederick Wilson(Father, non CS. Background with electronics):
Pretty much followed the overview.
He doesn’t think you should be able to inadvertently lose save progress. Prevent accidental deletion. (Dual execution deal: Dual confirmation if you delete a step):
Important: A display that confirms a positive result. A green light for successful execution.

2: Patrick Gatewood (Senior Computer Science Major at Virginia Tech):
If he had an assignment to write assembly language, use this to test it.
He also sees this as being a good debugger.

Should be able to step through. Halt execution. While halted, view contents of registers. If there’s a pointer, be able to see value. And be able to save the state, and reload it later. Export that state (send it to someone else)

The UI should be very clear to use. Guiding without a manual. Quickly load a saved state. Not too complex.

Should not be able to change values of registers. Doesn’t expect to be able to go back in instructions.

3: Erik Volinic. (Engineering Student. Has a very entertaining thought process. **I promise that this response was not meant as a joke.**)
Immediate reaction. Seeing it as data requisition.
Assumes data is being manipulated.
Uses this concept in seeing how electric impulses in a body would react to signals from the body and translate that to appendage movements. See how those impulses are sent out by the body. (He’s relating it to his prosthetic anatomy theory).

Relating to Neural networks. Simulate a process. (this program will not be that advanced)

Expects to see an object from point a to point b. Have an automatic or a manual.

4: Ethan Martin (cs student)
“I would load in a pre existing arm program, and run it through once. Then run a second time, if something goes wrong, step through at certain points” Similar to a debugger.
Should be able to load program into memory. Execute normally. Pause execution, move forward “n” number of steps at a time. Resume execution or prematurely stop. View current state of the virtual machine.

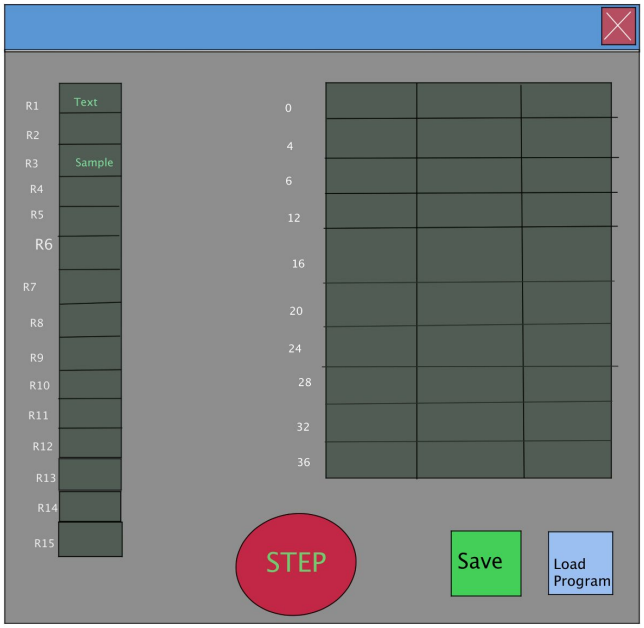
Should not be able to load programs that are not ARM related. Should not be able to step while program is executing in real time (while not halted).

Especially if this is a gui: All information he’d need being available. Cache, Ram, etc. See exact state.

User Model Summary:

ARRM is a GUI based ARM emulator that will guide the user through each step of the code’s execution. The interface will be very clear about how to use it as the buttons will be clearly labeled and verbose about their intended functions. The user will be able to open a file browser to pick out an ARM specific program file to load into the emulator. The program will iterate through the code and halt at a breakpoint until the user presses the “step” button. The user should not be able to step while the code execution is not halted.

4: Prototype



5: User Model Reactions:

1: Austin Tucker (non computer science):
He imagined using the load button first to load the program.
Use step to go step by step.
Save to save the state.

2: Paula Wilson (mom)
Literally the same responses as stated above.

3: Patrick Gatewood:

Would be useful if the registers were labeled as what they were (stack pointer).
Assumes he would use Load Program. Imagines seeing a not running/now running. Advance by using “Step”.
If you have a really long program, set a breakpoint instead step-step-step-step.
Be more verbose about what the user is “saving”.
Expects a file explorer.
Highlight the line you are on.

4: Ethan Martin:
Load Program, bringing up a file browser.
Does not have a real time run.

He imagines that he is mashing step the entire time.

6: Reflections

After considering the feedback given, I'd like to speak with my group as well as the client as to if there should be the option for unhalted execution and the ability to switch between that and a step by step execution. I will also make my buttons more verbose about how they are meant to be used. I also believe there should be an indicator (red light, green light) that signals if the code executed without an issue, or if an error has occurred.