Josh Fridstrom's Senior Project Proposal
10-22-15

Modtalk Debugger

Summary: For my senior project, I plan to design and implement a debugger for the Modtalk project. By the end of the semester I will be able to compile a Modtalk program for debug, run the program in debug mode, connect a debugger to the program and then set breakpoints, step through the code, inspect objects, show stack traces, switch processes, and change the value of variables while the program is running.

The project will consist of the following parts each worth the specified points.

1. Bytecode interpreter	**14**
I will need a bytecode interpreter for compact code sequences and so that I can easily slip in breakpoints. The bytecode interpreter will be partly written in C as part of the core runtime, but there will be a part written in Smalltalk that will generate C code to run on the runtime. Points will be awarded based on the kinds of programs can that be compiled and run with the bytecode interpreter.	
* Points for getting Nano to run (the smallest possible program)	2
* Points for getting Simple to run (a very simple program with modules)	5
* Points for getting ANSITester to run (tests all major components of the language)	5
* Points for getting DeltaBlue to run (we want to get a benchmark score)	2
2. Evaluation control protocol design	**10**
I will document the initial design of the evaluation control protocol and keep the documentation updated as I make discoveries and changes during implementation. Evaluation control protocol will specify how the debugger connects to the running program and what the communication between them looks like. Part of the design will be determining what happens when a breakpoint is hit. On the first implementation, when a breakpoint is hit, all other processes will halt and the control of the program will be given to the debugger process.	
* Points for initial draft	5
* Points for having an updated document at the end	5
3. Evaluation control implementation between processes w/shared queues	**33**
In the first implementation, the debugger will be a different processes in the same program that is under development. The debugger will communicate with the the other processes via a shared queue. This implementation will validate the protocol design.	
* Points for creating a pluggable debugger subsystem	1
* Points for evaluation control commands (step, resume)	10
* Points for getting a stack trace	2

* Points for accessing temps are arguments in a method 5
* Points for accessing permanent objects 5
* Points for accessing runtime allocated objects. 5
* Points for changing the value of a variable 5

4. Debugger control interface design **10**

I will keep a document of the initial design of the debugger control interface and keep the document updated as I make changes during implementation. Debugger control interface will specify what commands a user can give to the debugger in order to control a the evaluation of a debugged program.

* Points for the initial draft 5
* Points for having an updated document at the end 5

5. Decompiler **10**

I will need a decompiler for my command line debugger to show a user the source code for a method they are currently breaking in.

* Points for producing source code for a method without blocks 5
* Points for handling blocks and showing method code 3
* Points for showing where you are in the method 2

6. Debugger control command line interface **23**

The first implementation of the debugger control interface will be via the command line. A user can run a program compiled for debug and issue commands similar to gdb in order to debug their program.

* Use io controls to make async fd 5
* Set sigio handler to signal interrupt into the program 5
* Program interrupt handler will signal appropriate semaphores 3
* Points for setting/removing breakpoints on the entry to a method 3
* Points for stepping through a method and resuming 3
* Points for dumping the stack 3
* Points for showing the decompiled method currently breaking in 3
* Points for setting/removing breakpoints on a message send 3

Total Points **100**

Extra credit opportunities

7. Remote object framework **10**

I need a remote object framework implementation in Modtalk so that my debugger and program under debug can communicate via message sends. Eventually, I would like the debugger to communicate with the program over a socket.

8. Evaluation control protocol implementation #2 **10**

Better debugging for multi-process programs. Processes

not currently being debugged will continue to run when
the debugged process hits a breakpoint.

9. Evaluation control protocol implementation #3 **5**

Switch to using sockets instead of shared queues.

10. Debugger control interface implementation #2 **10**

The implementation of the debugger control interface will integrate with the existing Modtalk IDE. A user will be able to set breakpoints, view stack traces, inspect objects, and control the program evaluation all from the IDE.

* Points for turning position in string to position in code stream
* Points for coloring where a breakpoint is set in the IDE
* Points for adding menu to toggle breakpoints on message sends
* Points for the stack and inspector guis
* Points for being able to step through code when a breakpoint is hit

A: 90-100
B: 80-89
C: 70-79
D: 60-69
E: <60