iWarp DRAM Utilization

Bill Shubert

1.0 Current iWarp Memory Layout

Currently, the iWarp memory layout is a single flat address space, all static RAM. When applications are loaded, they are placed in low memory. The stack begins at high memory and grows down, while the heap begins immediately after the application text, data, and BSS sections and grows up.

Current Application Memory Layout

This is not adequate for a system including dynamic ram. It would atomatically place stack in dynamic ram, while as much application as possible would be placed in the static ram. Many application may have special needs with respect to the dynamic ram. For example, an application which makes heavy use of stack but rarely accesses it's data segment may wish to put data at the end of memory and put the stack in static ram.

2.0 Memory layout with DRAM

With the dynamic ram boards installed memory is divided into two regions; the static ram, which is faster to access, and the dynamic ram, which is slower but larger. Depending on the machine configuration, there may be a gap in the address space between the two memory banks where no memory will exist. This makes it necessary to consider the two types of memory as two non-contiguous segments.

Unlike the SRAM-only memory arrangement, with DRAM present it is no longer advantageous to keep the application data space in one block. In addition, with two memory blocks it is necessary to have two heaps and two stacks.

DRAM Application Memory Layout

Static Ram		
Application Static Block	Static Heap →	← Static stack

Dynamic Ram		
Application Dynamic Block	Dynamic Heap →	← Dynamic stack

3.0 User Interface

3.1 Application program interface

For an application to make maximum possible use of the two types of RAM provided by the DRAM module, it is necessary that there be a way for programs to request a specific type of memory during a call to `malloc()`. The standard call to `malloc()` will work as normal, trying to allocate memory from the default heap (which is set during load time as specified below). If the default heap is full, memory will be allocated out of the secondary heap. An additional call will be provided to automatically allocate memory from either segment of RAM.

During loading, the programmer will be able to select a default stack in either the static or dynamic RAM. The `sphi` and `splo` control space registers will be set appropriately to trigger an interrupt when the stack leaves the current stack frame. The stack will then continue in the secondary stack area. This will require no application code.

3.2 Loading interface

Switches will be provided to the `iwld` command to enable the programmer to select, at loading time, which memory block to place the five parts of the program - text, data, BSS, heap, and stack. For any program part whose location is not specified in the `iwld` command line, a heuristic will be applied in an attempt to locate the program part as advantageously as possible.

New switches for `iwld`:

`-ts`, `-td` Text in static or dynamic RAM

`-ds`, `-dd`	Data in static or dynamic RAM
`-bs`, `-bd`	BSS in static or dynamic RAM
`-hs`, `-hd`	Default heap in static or dynamic RAM
`-ss`, `-sd`	Default stack in static or dynamic RAM

4.0 Performance Implications

Unfortunately the DRAM boards are not yet ready for use so no performace testing has been done. The heuristic mentioned in 3.2 will also have to wait until numbers are obtained.