Determining the Remaining Stack Space on HP-UX Threads Running on the Intel® Itanium® Architecture

Challenge
Determining how much stack space remains in the current thread.

Solution
Using the inline assembler and some of the built-in assembly pseudo functions, read the stack pointer and backing store pointer and compare their values. The Intel Itanium processor actually has two stacks, the memory stack, which is the conventional stack used directly by the compiler to store local variables and temporaries and the register stack, which is used by the processor as backing store for the rotating register file. The memory stack grows down while the register stack grows up. By convention on HP-UX systems, these stacks are allocated adjacent, with the stack pointers growing together. Typically, an un-writeable guard page is placed between the two stacks.

Provided the stack space was allocated using the default scheme, computing the remaining stack space involves subtracting the current stack pointer (SP) from the current backing store pointer (BSP). Both the initial thread (thread 0) allocated by the kernel and subsequent threads allocated by libpthread will allocate stacks with the memory stack directly above the register stack.

Using the inline assembly capabilities build into the C and C++ compilers, the remaining stack space can be determined with a function similar to:

```c
#include <sys/types.h>
#include <machine/sys/inline.h>

#define GUARD_PAGE_SIZE 4096

size_t
stack_space_remaining()
{
    intptr_t sp, bsp;
    sp = (intptr_t)_Asm_get_sp();
    bsp = (intptr_t)_Asm_mov_from_ar(_AREG_BSP);
    if (sp > bsp)
        return ((size_t)(sp - bsp - GUARD_PAGE_SIZE));
    else
        return 0;
}
```

If the stack pointer is less than the backing store pointer, it is likely that the stacks were not allocated adjacent to each other. The size of the guard page should be adjusted to correspond to values set via `pthread_attr_setguardsize()`.

It is important to note that this method will determine the amount of total stack space left and not necessarily the space available to either the memory stack or register stack due to the placement of the guard page. If the guard page size is set to 0 using `pthread_attr_setguardsize()`, then libpthread will allocate stack space without a guard page. This eliminates the safety net of a guard page for stack overflow, but it also allows the stack space to be used in a flexible manner by both the memory stack and
register stack. Without a guard page, this method will return the combined stack space available to the calling thread.

**Source**

[Inline assembly for Itanium-based HP-UX](#)