CLAIMS

1. A method of implementing protection key memory access control in a computing system having a processor not adapted to support protection key memory access control, the processor including a memory management unit (MMU) and executing an operating system that manages virtual memory, the processor adapted to execute user processes comprising tasks, the method comprising:

assigning a contiguous range of addressable virtual memory containing blocks to each user process;

assigning a storage access key to each task;

initiating execution of a specific task in a specific user process using the storage access key assigned to that specific task by assigning a first virtual address mapping in the contiguous range of addressable virtual memory to that specific task with its assigned storage access key;

in response to a subsequent memory access by the specific task, if the subsequent memory access uses a storage access key different from any storage access key previously used by the specific task, determining whether the specific task is authorized to use the different storage access key, and if the specific task is authorized, assigning a subsequent virtual address mapping to the specific task and the different storage access key;

in response to a command to allocate or to free storage used by the specific task, setting control bytes including setting storage key data and an indicator that a page comprising data associated with each control byte is valid; and

upon execution of a computer instruction of the specific task using any storage access key,

generating a virtual address associated with the specific task;

determining whether the virtual address associated with the specific task lies within the contiguous range of virtual address memory; and

if the address lies within the contiguous range of addressable virtual memory:

generating by the MMU a segmentation fault;

verifying by an exception handler that said storage access key has a value of zero or that said task is authorized to access the virtual address using that said storage access key;

upon said verifying an exception handler, changing native protection settings associated with the page stored in the virtual memory and associated with the task to allow access to the associated virtual address; and

subsequently re-executing the computer instruction of the specific task to allow access to the page.

2. A method of implementing protection key memory access control in a computing system having a processor not adapted to support protection key memory access control, the processor including a memory management unit (MMU) and executing an operating system that manages virtual memory, the processor adapted to execute user processes comprising tasks, the method comprising:

assigning a contiguous range of addressable virtual memory containing blocks to each user process;

assigning a storage access key to each task;

initiating execution of a specific task in a specific user process using the storage access key assigned to that specific task by assigning a first virtual address mapping in the contiguous range of addressable virtual memory to that specific task with its assigned storage access key;

determining whether the specific task is authorized to use a different storage access key, and if the specific task is authorized, assigning a subsequent virtual address mapping to the specific task and the different storage access key;

setting control bytes including setting storage key data and an indicator that a page comprising data associated with each control byte is valid; and

upon execution of a computer instruction of the specific task using any storage access key,

generating a virtual address associated with the specific task;

determining whether the virtual address associated with the specific task lies within the contiguous range of virtual address memory; and

if the address lies within the contiguous range of addressable virtual memory:

generating by the MMU a segmentation fault;

verifying that said storage access key has a value of zero or that said task is authorized to access the virtual address using that said storage access key;

upon said verifying, changing native protection settings associated with the page stored in the virtual memory and associated with the task to allow access to the associated virtual address; and

subsequently re-executing the computer instruction of the specific task to allow access to the page.

3. The method of claim 1 or 2, wherein the blocks comprise 4K blocks.

4. The method of claim 1 or 2, wherein the segmentation fault comprises indicating that the page corresponding to the virtual address is not present in physical memory.

5. The method of claim 1 or 2, wherein the segmentation fault comprises indicating that the task is attempting to write to the page which is present in memory with a read only access permission setting.

6. The method of claim 1 or 2, wherein the segmentation fault comprises indicating that the task is attempting to read the page without a matching access permission setting.

7. The method of claim 1 or 2, wherein prefix data associated with the first and subsequent executions of the specific task of the specific user process is stored at the same logical address, but in different physical addresses.

8. The method of claim 1 or 2, wherein changing native protection settings associated with the page stored in the virtual address memory and associated with the

task comprises setting the value of bit zero of a corresponding page table entry to indicate that the physical page comprising the data is present in memory.

9. The method of claim 1 or 2, further comprising invoking the MPROTECT( ) function to change native protection settings associated with the page stored in the virtual address memory and associated with the specific task.

10. The method of claim 1 or 2, wherein changing native protection settings comprises changing access control bits.

11. The method of claim 1 or 2, further comprising, in response to the segmentation fault, executing a routine to change a state of the page stored in the virtual address memory and associated with the specific task from not-present to present.

12. A computing system comprising: a processor not adapted to support protection key memory access control, the processor including a memory management unit (MMU) and executing an operating system that manages addressable virtual memory, the processor adapted to execute user processes comprising tasks; and

a non-transient memory storing instructions which, when executed on the processor, cause the processor to:

assign to a user process, a contiguous range of addressable virtual memory containing blocks and assign a storage key to each block;

assign a storage access key to each task;

initiate execution of a specific task in a specific user process using a specific storage access key assigned to that specific task by assigning a first virtual address mapping in the contiguous range of addressable virtual memory to that specific task with its assigned storage access key;

in response to a subsequent memory access by the specific task, if the subsequent memory access uses a storage access key different from any storage access keys previously used by the specific task, determine whether the specific task is authorized to use the different storage access key, and if the specific task is

authorized, assign a subsequent virtual address mapping to the specific task and different storage access key;

in response to a command to allocate or to free storage by the specific task, set control bytes including setting the storage key data and an indicator that a page comprising data associated with each control byte is valid; and

upon execution of a computer instruction of the specific task using any storage access key:

generate a virtual address in the address associated with the specific task;

determine whether the virtual address associated with the specific task lies within the range of contiguous range of addressable virtual memory; and

if the address lies within the contiguous range of addressable virtual memory:

generate by the MMU a segmentation fault;

verify by an exception handler that said storage access key has a value of zero or that said task is authorized to access the virtual address in the addressable virtual memory using that said storage access key;

upon said verifying an exception handler, change native protection settings associated with the page stored in the virtual memory and associated with the task to allow access to the associated virtual address; and

subsequently re-execute the computer instruction of the specific task to allow access to the page.

13. A computing system comprising: a processor not adapted to support protection key memory access control, the processor including a memory management unit (MMU) and executing an operating system that manages addressable virtual memory, the processor adapted to execute user processes comprising tasks; and

a non-transient memory storing instructions which, when executed on the processor, cause the processor to:

assign to a user process, a contiguous range of addressable virtual memory containing blocks and assign a storage key to each block;

assign a storage access key to each task;

initiate execution of a specific task in a specific user process using a specific storage access key assigned to that specific task by assigning a first virtual address mapping in the contiguous range of addressable virtual memory to that specific task with its assigned storage access key;

determine whether the specific task is authorized to use a different storage access key, and if the specific task is authorized, assign a subsequent virtual address mapping to the specific task and different storage access key;

set control bytes including setting the storage key data and an indicator that a page comprising data associated with each control byte is valid; and

upon execution of a computer instruction of the specific task using any storage access key:

generate a virtual address in the address associated with the specific task;

determine whether the virtual address associated with the specific task lies within the range of contiguous range of addressable virtual memory; and

if the address lies within the contiguous range of addressable virtual memory:

generate by the MMU a segmentation fault;

verify that said storage access key has a value of zero or that said task is authorized to access the virtual address in the addressable virtual memory using that said storage access key;

upon said verifying, change native protection settings associated with the page stored in the virtual memory and associated with the task to allow access to the associated virtual address; and

subsequently re-execute the computer instruction of the specific task to allow access to the page.

14. The system of claim 12 or 13, wherein the blocks comprise 4K blocks.

15. The system of claim 12 or 13, wherein the segmentation fault comprises and indication that the page corresponding to the virtual address is not present in physical memory.

16. The system of claim 12 or 13, wherein the segmentation fault comprises an indication that the task is attempting to write to the page which is present in memory with a read only access permission setting.

17. The system of claim 12 or 13, wherein the segmentation fault comprises an indication that the task is attempting to read the page without a matching access permission setting.

18. The system of claim 12 or 13, wherein prefix data associated with the first and subsequent executions of the specific task of the specific user process is stored at the same logical address, but in different physical addresses.

19. The system of claim 12 or 13, wherein to change native protection settings associated with the page stored in the virtual address memory and associated with the task, the value of bit zero of a corresponding page table entry is set to indicate that the physical page comprising the data is present in memory.

20. The system of claim 12 or 13, further comprising the non-transient memory storing instructions which, when executed on the processor, cause the processor to invoke the MPROTECT( ) function to change native protection settings associated with the page stored in the virtual address memory and associated with the specific task.

21. The system of claim 12 or 13, wherein the instructions cause the processor to change native protection settings by changing access control bits.

22. The system of claim 12 or 13, further comprising, in response to the segmentation fault, executing a routine to change a state of the page stored in the

virtual address memory and associated with the specific task from not-present to present.