CLAIMS

1. A computer program product comprising a computer readable storage medium readable by a processing circuit and storing program code for execution by the processing circuit for performing a method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field to be used to indicate a starting address in main memory; and

a block boundary size indicator to indicate a block boundary size of a block of main memory;

and

executing the machine instruction, the executing comprising:

computing the starting address from the at least one field of the machine instruction, the starting address indicating a location in main memory from which loading into the first operand is to begin;

computing an ending address in main memory at which loading into the first operand is to stop, wherein the computing the ending address includes using the following equation: ending address= minimum of (starting address + (boundary size- (starting address AND NOT boundary mask)), starting address+ register size), wherein the boundary size is the block boundary size, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register designated in the register field of the machine instruction; and loading a variable amount of data from main memory into the first operand, the loading comprising loading from main memory starting at the starting address in main memory and terminating at the ending address in main memory, wherein the variable amount of data is loaded from main memory into the first operand without crossing a designated main memory boundary.

2. The computer program product of claim 1, wherein the at least one field comprises a displacement field, a base field and an index field, the base field and index field for locating

general registers having contents to be added to contents of the displacement field to form the starting address, and wherein the machine instruction further comprises a mask field, the mask field specifying the block boundary size indicator.

3. The computer program product of claim 2, wherein the block boundary size is one block boundary size of a plurality of block boundary sizes specifiable by the mask field.

4. The computer program product of any one of the preceding claims, wherein the loading comprises one of: loading the first operand from left to right, or loading the first operand from right to left.

5. The computer program product of claim 4, wherein a direction of the loading is provided at runtime.

6. The computer program product of any one of the preceding claims, wherein the machine instruction further comprises an extension field to be used in designating one or more registers, and wherein the register field is combined with at least a portion of the extension field to designate the register.

7. The computer program product of any one of the preceding claims, wherein the loading the first operand begins at an indexed byte 0 of the first operand, and continues up to a maximum byte index of the register.

8. A computer system comprising:

a memory; and

a processor in communications with the memory, wherein the computer system is configured to perform a method, said method comprising:

obtaining, by the processor, a machine instruction for execution, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation; a register field to be used to designate a register, the register comprising a first operand; at least one field to be used to indicate a starting address in main memory; and

a block boundary size indicator to indicate a block boundary size of a block of main memory; and

executing the machine instruction, the executing comprising:

computing the starting address from the at least one field of the machine instruction, the starting address indicating a location in main memory from which loading into the first operand is to begin;

computing an ending address in main memory at which loading into the first operand is to stop, wherein the computing the ending address includes using the following equation: ending address= minimum of (starting address+ (boundary size- (starting address AND NOT boundary mask)), starting address+ register size), wherein the boundary size is the block boundary size, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register designated in the register field of the machine instruction; and loading a variable amount of data from main memory into the first operand, the loading comprising loading from main memory starting at the starting address in main memory and terminating at the ending address in main memory, wherein the variable amount of data is loaded from main memory into the first operand without crossing a designated main memory boundary.

9. The computer system of claim 8, wherein the at least one field comprises a displacement field, a base field and an index field, the base field and index field for locating general registers having contents to be added to contents of the displacement field to form the starting address, and wherein the machine instruction further comprises a mask field, the mask field specifying the block boundary size indicator.

10. The computer system of any one of claims 8 to 9, wherein the loading comprises one of:

loading the first operand from left to right, or loading the first operand from right to left.

11. The computer system of claim 10, wherein a direction of the loading is provided at runtime.

12. The computer system of any one of claims 8 to 11, wherein the machine instruction further comprises an extension field to be used in designating one or more registers, and wherein the register field is combined with at least a portion of the extension field to designate the register.

13. A method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field to be used to indicate a starting address in main memory; and

a block boundary size indicator to indicate a block boundary size of a block of main memory; and

executing the machine instruction, the executing comprising:

computing the starting address from the at least one field of the machine instruction, the starting address indicating a location in main memory from which loading into the first operand is to begin;

computing an ending address in main memory at which loading into the first operand is to stop, wherein the computing the ending address includes using the following equation: ending address= minimum of (starting address+ (boundary size- (starting address AND NOT boundary mask)), starting address+ register size), wherein the boundary size is the block boundary size, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register designated in the register field of the machine instruction; and

loading a variable amount of data from main memory into the first operand, the loading comprising loading from main memory starting at the starting address in main memory and terminating at the ending address in main memory, wherein the variable amount of data is loaded from main memory into the first operand without crossing a designated main memory boundary.

CLAIMS

1. A computer program product for executing a machine instruction in a central processing unit, the computer program product comprising:

a computer readable storage medium readable by a processing circuit and storing instructions for execution by the processing circuit for performing a method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field for locating a second operand in main memory; and

executing the machine instruction, the execution comprising:

only loading bytes of the first operand with corresponding bytes of the second operand that are within an instruction specified block of main memory.

2. The computer program product of claim 1, wherein the at least one field comprises a displacement field, a base field and an index field, the base field and index field for locating general registers having contents to be added to contents of the displacement field to form an address of the second operand, and wherein the machine instruction further comprises a mask field, the mask field specifying the block boundary.

3. The computer program product of claim 2, wherein the block boundary is one block boundary of a plurality of block boundaries specifiable by the mask field.

4. The computer program product of any preceding claim, wherein the address of the second operand is a starting address in memory from which data is to be loaded in the first operand.

5. The computer program product of claim 4, wherein the executing further comprises determining an ending address at which loading is to stop, and wherein the loading stops at the ending address.

6. The computer program product of claim 5, wherein the determining the ending address comprises computing the ending address as follows:

ending address = minimum of (starting address + (boundary size – (starting address AND boundary mask)), starting address + register size), wherein the boundary size is the block boundary, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register.

7. The computer program product of any preceding claim, wherein the loading comprises one of: loading the first operand from left to right, or loading the first operand from right to left.

8. The computer program product of claim 7, wherein a direction of the loading is provided at runtime.

9. The computer program product of any preceding claim, wherein the machine instruction further comprises an extension field to be used in designating one or more registers, and wherein the register field is combined with at least a portion of the extension field to designate the register.

10. The computer program product of any preceding claim, wherein the loading the first operand begins at an indexed byte 0 of the first operand, and continues up to a maximum byte index of the register.

11. A computer system for executing a machine instruction in a central processing unit, the computer system comprising:

a memory; and

a processor in communications with the memory, wherein the computer system is configured to perform a method, said method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field for locating a second operand in main memory; and

executing the machine instruction, the execution comprising:

only loading bytes of the first operand with corresponding bytes of the second operand that are within an instruction specified block of main memory.

12. The computer system of claim 11, wherein the at least one field comprises a displacement field, a base field and an index field, the base field and index field for locating general registers having contents to be added to contents of the displacement field to form an address of the second operand, and wherein the machine instruction further comprises a mask field, the mask field specifying the block boundary.

13. The computer system of claim 11 or claim 12, wherein the address of the second operand is a starting address in memory from which data is to be loaded in the first operand, and wherein the executing further comprises determining an ending address at which loading is to stop, and wherein the loading stops at the ending address.

14. The computer system of claim 13, wherein the determining the ending address comprises computing the ending address as follows:

ending address = minimum of (starting address + (boundary size – (starting address AND boundary mask)), starting address + register size), wherein the boundary size is the block boundary, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register.

15. The computer system of any of claims 11 to 14, wherein the loading comprises one of: loading the first operand from left to right, or loading the first operand from right to left.

16. The computer system of claim 15, wherein a direction of the loading is provided at runtime.

17. The computer system of any of claims 11 to 16, wherein the machine instruction further comprises an extension field to be used in designating one or more registers, and wherein the register field is combined with at least a portion of the extension field to designate the register.

18. A method of executing a machine instruction in a central processing unit, the method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field for locating a second operand in main memory; and

executing the machine instruction, the execution comprising:

only loading bytes of the first operand with corresponding bytes of the second operand that are within an instruction specified block of main memory.

19. The method of claim 18, wherein the address of the second operand is a starting address in memory from which data is to be loaded in the first operand, and wherein the executing further comprises determining an ending address at which loading is to stop, and wherein the loading stops at the ending address.

20. The method of claim 19, wherein the determining the ending address comprises computing the ending address as follows:

ending address = minimum of (starting address + (boundary size – (starting address AND boundary mask)), starting address + register size), wherein the boundary size is the block boundary, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register.

AMENDED CLAIMS
received by the International Bureau on 5 July 2013 (05.07.2013).

1. A computer program product for executing a machine instruction in a central processing unit, the computer program product comprising a computer readable storage medium readable by a processing circuit and storing instructions for execution by the processing circuit for performing a method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field for locating a second operand in main memory;

a block boundary size indicator to indicate a specified block boundary of a block of the second operand; and

executing the machine instruction comprising: loading from a block of the second operand a variable amount of data into the first operand while ensuring only data within the block is being loaded into the first operand, wherein the loading starts at a starting address in the block of the second operand, the starting address provided by the machine instruction, wherein the loading ends at or before the specified block boundary of the block of the second operand, wherein the variable amount of data loaded is based on the starting address and the specified block boundary, the specified block boundary determined based on the block boundary size indicator, and wherein the variable amount of data is a minimum of (a number of bytes in the first operand or a number of bytes loaded up to the specified block boundary).

2. The computer program product of claim 1, wherein the at least one field comprises a displacement field, a base field and an index field, the base field and index field for locating general registers having contents to be added to contents of the displacement field to form an address of the second operand, and wherein the machine instruction further comprises a mask field, the mask field specifying the block boundary size indicator.

3. The computer program product of claim 2, wherein the block boundary is one block boundary of a plurality of block boundaries specifiable by the mask field.

4. The computer program product of any preceding claim, wherein the address of the second operand is a starting address in memory from which data is to be loaded in the first operand.

5. The computer program product of claim 4, wherein the executing further comprises determining an ending address at which loading is to stop, and wherein the loading stops at the ending address.

6. The computer program product of claim 5, wherein the determining the ending address comprises computing the ending address as follows:

ending address = minimum of (starting address + (boundary size – (starting address AND boundary mask)), starting address + register size), wherein the boundary size is the block boundary, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register.

7. The computer program product of any preceding claim, wherein the loading comprises one of: loading the first operand from left to right, or loading the first operand from right to left.

8. The computer program product of claim 7, wherein a direction of the loading is provided at runtime.

9. The computer program product of any preceding claim, wherein the machine instruction further comprises an extension field to be used in designating one or more registers, and wherein the register field is combined with at least a portion of the extension field to designate the register.

10. The computer program product of any preceding claim, wherein the loading the first operand begins at an indexed byte 0 of the first operand, and continues up to a maximum byte index of the register.

11. A computer system for executing a machine instruction in a central processing unit, the computer system comprising:

a memory; and

a processor in communications with the memory, wherein the computer system is configured to perform a method, said method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field for locating a second operand in main memory;

a block boundary size indicator to indicate a specified block boundary of a block of the second operand; and

executing the machine instruction comprising:

loading from a block of the second operand a variable amount of data into the first operand while ensuring only data within the block is being loaded into the first operand, wherein the loading starts at a starting address in the block of the second operand, the starting address provided by the machine instruction, wherein the loading ends at or before the specified block boundary of the block of the second operand, wherein the variable amount of data loaded is based on the starting address and the specified block boundary, the specified block boundary determined based on the block boundary size indicator, and wherein the variable amount of data is a minimum of (a number of bytes in the first operand or a number of bytes loaded up to the specified block boundary).

12. The computer system of claim 11, wherein the at least one field comprises a displacement field, a base field and an index field, the base field and index field for locating

general registers having contents to be added to contents of the displacement field to form an address of the second operand, and wherein the machine instruction further comprises a mask field, the mask field specifying the block boundary size indicator.

13. The computer system of claim 11 or claim 12, wherein the address of the second operand is a starting address in memory from which data is to be loaded in the first operand, and wherein the executing further comprises determining an ending address at which loading is to stop, and wherein the loading stops at the ending address.

14. The computer system of claim 13, wherein the determining the ending address comprises computing the ending address as follows:

ending address = minimum of (starting address + (boundary size – (starting address AND boundary mask)), starting address + register size), wherein the boundary size is the block boundary, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register.

15. The computer system of any of claims 11 to 14, wherein the loading comprises one of: loading the first operand from left to right, or loading the first operand from right to left.

16. The computer system of claim 15, wherein a direction of the loading is provided at runtime.

17. The computer system of any of claims 11 to 16, wherein the machine instruction further comprises an extension field to be used in designating one or more registers, and wherein the register field is combined with at least a portion of the extension field to designate the register.

18. A method of executing a machine instruction in a central processing unit, the method comprising:

obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising:

at least one opcode field to provide an opcode, the opcode identifying a load to block boundary operation;

a register field to be used to designate a register, the register comprising a first operand;

at least one field for locating a second operand in main memory;

a block boundary size indicator to indicate a specified block boundary of a block of the second operand; and

executing the machine instruction comprising:

loading from a block of the second operand a variable amount of data into the first operand while ensuring only data within the block is being loaded into the first operand, wherein the loading starts at a starting address in the block of the second operand, the starting address provided by the machine instruction, wherein the loading ends at or before the specified block boundary of the block of the second operand, wherein the variable amount of data loaded is based on the starting address and the specified block boundary, the specified block boundary determined based on the block boundary size indicator, and wherein the variable amount of data is a minimum of (a number of bytes in the first operand or a number of bytes loaded up to the specified block boundary).

19. The method of claim 18, wherein the address of the second operand is a starting address in memory from which data is to be loaded in the first operand, and wherein the executing further comprises determining an ending address at which loading is to stop, and wherein the loading stops at the ending address.

20. The method of claim 19, wherein the determining the ending address comprises computing the ending address as follows:

ending address = minimum of (starting address + (boundary size – (starting address AND boundary mask)), starting address + register size), wherein the boundary size is the block boundary, boundary mask is equal to 0 – boundary size, and register size is a specified length of the register.