1. **Instruction set:** The vocabulary of commands understood by a given architecture.
2. **Stored-program concept:** The idea that instructions and data of many types can be stored in memory as numbers and thus be easy to change, leading to the stored-program computer.
3. **Doubleword:** Another natural unit of access in a computer, usually a group of 64 bits; corresponds to the size of a register in the RISC-V architecture.
4. **Word:** A natural unit of access in a computer, usually a group of 32 bits.
5. **Data transfer instruction:** A command that moves data between memory and registers.
6. **Address:** A value used to delineate the location of a specific data element within a memory array.
7. **Alignment restriction:** A requirement that data be aligned in memory on natural boundaries.
8. **Binary digit:** Also called bit. One of the two numbers in base 2, 0 or 1, that are the components of information.
9. **Least significant bit:** The rightmost bit in an RISC-V doubleword.
10. **Most significant bit:** The leftmost bit in an RISC-V doubleword.
11. **One's complement:** A notation that represents the most negative value by $10\cdots000_2$ and the most positive value by $01\cdots11_2$, leaving an equal number of negatives and positives but ending up with two zeros, one positive ($00\cdots00_2$) and one negative ($11\cdots11_2$). The term is also used to mean the inversion of every bit in a pattern: 0 to 1 and 1 to 0.
12. **Biased notation:** A notation that represents the most negative value by $00\cdots000_2$ and the most positive value by $11\cdots11_2$, with 0 typically having the value $10\cdots00_2$, thereby biasing the number such that the number plus the bias has a non-negative representation.
13. **Instruction format:** A form of representation of an instruction composed of fields of binary numbers.
14. **Machine language:** Binary representation used for communication within a computer system.
15. **Hexadecimal:** Numbers in base 16.
16. **Opcode:** The field that denotes the operation and format of an instruction.
17. **Basic block:** A sequence of instructions without branches (except possibly at the end) and without branch targets or branch labels (except possibly at the beginning).
18. **Branch address table:** Also called branch table. A table of addresses of alternative instruction sequences.
19. **Procedure:** A stored subroutine that performs a specific task based on the parameters with which it is provided.
20. **Jump-and-link instruction:** An instruction that branches to an address and simultaneously saves the address of the following instruction in a register (usually x1 in RISC-V).
21. **Return address:** A link to the calling site that allows a procedure to return to the proper address; in RISC-V it is stored in register x1.
22. **Caller:** The program that instigates a procedure and provides the necessary parameter values.
23. **Callee:** A procedure that executes a series of stored instructions based on parameters provided by the caller and then returns control to the caller.
24. **Program counter (PC):** The register containing the address of the instruction in the program being executed.

25. **Stack:** A data structure for spilling registers organized as a last-in-first-out queue.

26. **Stack pointer:** A value denoting the most recently allocated address in a stack that shows where registers should be spilled or where old register values can be found. In RISC-V, it is register sp, or x2.

27. **Global pointer:** The register that is reserved to point to the static area.

28. **Procedure frame:** Also called activation record. The segment of the stack containing a procedure's saved registers and local variables.

29. **Frame pointer:** A value denoting the location of the saved registers and local variables for a given procedure.

30. **Text segment:** The segment of a UNIX object file that contains the machine language code for routines in the source file.

31. **PC-relative addressing:** An addressing regime in which the address is the sum of the program counter (PC) and a constant in the instruction.

32. **Addressing mode:** One of several addressing regimes delimited by their varied use of operands and/or addresses.

33. **Data race:** Two memory accesses form a data race if they are from different threads to the same location, at least one is a write, and they occur one after another.

34. **Assembly language:** A symbolic language that can be translated into binary machine language.

35. **Pseudoinstruction:** A common variation of assembly language instructions often treated as if it were an instruction in its own right.

36. **Symbol table:** A table that matches names of labels to the addresses of the memory words that instructions occupy.

37. **Linker:** Also called link editor. A systems program that combines independently assembled machine language programs and resolves all undefined labels into an executable file.

38. **Executable file:** A functional program in the format of an object file that contains no unresolved references. It can contain symbol tables and debugging information. A "stripped executable" does not contain that information. Relocation information may be included for the loader.

39. **Loader:** A systems program that places an object program in main memory so that it is ready to execute.

40. **Dynamically linked libraries (DLLs):** Library routines that are linked to a program during execution.

41. **Java bytecode:** Instruction from an instruction set designed to interpret Java programs.

42. **Java Virtual Machine (JVM):** The program that interprets Java bytecodes.

43. **Just In Time compiler (JIT):** The name commonly given to a compiler that operates at runtime, translating the interpreted code segments into the native code of the computer.