Assembly Glossary

Created By: Nafeel Ahmed, Teaching Assistant

1. **Assembler -** Assembler often abbreviated asm, is a program for converting instructions written in low-level assembly code into relocatable machine code and generating along information for the loader.
2. **Compiler -** A compiler is a computer program that translates computer code written in one programming language into another language.
3. **MASM -** Microsoft Assembler.
4. **TASM -** Turbo Assembler.
5. **NASM -** Netwide Assembler.
6. **GNU Assembler -** The GNU Assembler, commonly known as gas or simply as, its executable name, is the assembler used by the GNU Project.
7. **ISA -** An instruction set architecture is an abstract model of a computer.
8. **Nibble -** 4 Bits.
9. **Byte -** 8 Bits.
10. **MSB -** The most significant bit (MSB, also called the high-order bit) is the bit position in a binary number having the greatest value.
11. **ASCII -** American Standard for Information Interchange.
12. **UNICODE -** One standard for all of the world.
13. **Registers -** Registers are a type of computer memory used to quickly accept, store, and transfer data and instructions that are being used immediately by the CPU.
14. **Flags -** The FLAGS register is the status register in Intel x86 microprocessors that contains the current state of the processor.
15. **Memory Address -** A memory address is a unique identifier used by a device or CPU for data tracking.
16. **IO Function -** The term I/O is used to describe any program, operation or device that transfers data to or from a computer and to or from a peripheral device.
17. **CPU -** Central Processing Unit.
18. **ALU -** Arithmetic Logic Unit.
19. **FLU -** Floating Point Unit.

20. **EAX -** 32 Bits, Accumulator and Returns .
21. **AX -** low 16 Bits of EAX.
22. **AL -** low 8 Bits of AX or EAX.
23. **AH -** high 8 bits of AX.
24. **ECX -** Counter.
25. **ESP -** Extended Stack Pointer.
26. **ESI -** Source.
27. **EDI -** Destination.
28. **EPB -** Extended Base Pointer, Frame pointer (when we use stack).
29. **EDX:EAX -** Multiplication and Divide.
30. **IP/EIP -** Extended Instruction Pointer.
31. **CF -** Carry.
32. **OF -** Overflow.
33. **SF -** Sign Flag.
34. **ZF -** Zero Flag.
35. **AC -** Auxiliary Carry.
36. **PF -** Parity.
37. **CS -** Code Segment.
38. **DS -** Data Segment.
39. **SS -** Stack Segment.
40. **MUL -** Multiply.
41. **MOV -** Move.
42. **DIV -** Divide.
43. **JMP -** Jump Location (Unconditional).
44. **JZ -** Conditional Jump.
45. **LOOP -** Decrements ecx.
46. **Shifting -** shr, Logical Shift, Move all the bits to the left or Right.
47. **Arithmetic Shift/Signed Shift -** SHLD/SHRD dest.
48. **Mnemonic -** Instruction To be executed.
49. **Operand -** A parameter to the instruction.
50. **SAL -** Shift Arithmetic Left.
51. **call funcation_Name -** Pushes the address of the next instruction onto the stack.
52. **ret -** Return from a function.
53. **Sub esp, X -** Subtract X bytes from esp.

54. **Add esp, X -** Add X bytes to esp.
55. **And esp, 0FFFFFFF0h -** To make sure that esp is on an even byte boundary.
56. **Printf** - C function, used to print statements.
57. **GEF -** GDB Enhanced Features.
58. **Function Prologue -** Setup entering a function.
59. **Function Epilogue -** To Restore the stack.
60. **Pushad -** Save all general purpose registers.
61. **Popad -** pop EDI,ESI,EBP,EBX,EDX.
62. **Calling Convention -** Assumptions that software makes about how the caller and callee work.
63. **Cdecl -** Call Declaration.
64. **Enter 0,0 -** Enter a function and create a stack frame.
65. **FADD -** Addition.
66. **FSUB -** Subtraction.
67. **FMUL -** Multiplication.
68. **FDIV -** Division.
69. **FCOM -** COMPARES SRC and ST0.
70. **FCOMP -** COMPARES SRC and ST0 and pops stack.
71. **FCOMPP -** Compares ST0 and ST1 and pop stack twice.
72. **FICOM -** Compare ST0 and src(int).
73. **FICOMP -** Compare ST0 and src(int) and then pop stack.
74. **FIST -** Compare ST0 and 0.
75. **FCHS -** Changes sign of ST0.
76. **FABS -** Absolute value of ST0.
77. **FSQRT -** Square root of ST0.
78. **FSCALE -** Exponent ST0 = ST0 x 2ST1.
79. **FCOS/FSIN/FPTAN -** Floating point cosine/Floating point sin/Floating point tan.
80. **Pipelines -** allows multiple instruction to be in the stages of processing.
81. **Branching -** Jumping.
82. **Conditional Execution -** The Ability to execute the instructions based on flags .
83. **Pointers -** Points to a location in memory(RAM).
84. **Arrays -** A region of contiguous memory, a consecutive sequence of bytes.
85. **Mov -** copies Address.
86. **Lea -** does Calculation,but does not load memory **.**

87. **ARM -** Acorn RISC Machine, later Advanced RISC Machine.
88. **CPSR -** Current Process Status Register (32 bits).
89. **RO-R15 -** R0 Accumulator. R13 is SP(Stack Pointer), R14 is LR(Link Register), R15 is PC(Program Counter).
90. **Banked Registers -** Banked Registers are discrete physical registers in the core that are mapped to the available registers depending on the current processor operating mode. Banked Register contents are preserved across operating mode changes.
91. **ABI -** Application Binary Interface.
92. **Ldm -** Load Multiple.
93. **Stm -** Store Multiple.
94. **IA -** Increment After.
95. **IB -** Increment Before.
96. **DA -** Decrement After.
97. **DB -** Decrement Before.
98. **VFP -** Vector Floating Registers .
99. **Neon -** Advanced SIMD for ARM.
100. **FPSCR -** Floating Point Status and Control Registers.
101. **VCVT -** v convert, Between single-precision and double precision resistors.
102. **SIMD -** Single Instruction Multiple Data.
103. **VLD -** v load.
104. **VSTR -** V store data.
105. **ARM Mode -** Instructions in 32 bits wide.
106. **Thumb Mode -** Instructions in 16 bits wide.
107. **BLX -** BRANCH with Link and exchange.
108. **UAL -** Unified Assembler Language.
109. **Conditional Execution -** Allows an instruction to be executed only when a condition is met.
110. **Thumb-2 Conditional Execution -** Does not have the bits for stand alone conditional instructions.
111. **IT Blocks - If Then** blocks.
112. **COFF -** Common Object File Format.
113. **PE Format -** Portable Executable Format.
114. **Union -** A union is a value that may have any of several representations or formats within the same position in memory.

115. **Structure (struct) -** A struct in the C programming language (and many derivatives) is a composite data type (or record) declaration that defines a physically grouped list of variables under one name in a block of memory, allowing the different variables to be accessed via a single pointer or by the struct declared name which returns the same address .
116. **Malloc -** Memory Allocation.
117. **Switch Statements -** Allows a programmer to easily choose a block of code to execute
118. **Jump Tables -** An array of pointers to program code.
119. **Function Pointer -** Points to a function or subroutine.
120. **Inline Assembly -** In computer programming, an inline assembler is a feature of some compilers that allows low-level code written in assembly language to be embedded within a program, among code that otherwise has been compiled from a higher-level language such as C or Ada.
121. **GCC -** GNU Compiler Collection.
122. **Reverse engineering-** It involves finding out how various functions in the code are built, what they do, and how each relates to and interacts with other code functions.
123. **Interrupts -** An event that signals to the CPU that something occurred.
124. **SysCall -** System Calls, Provides a way to interact with the operating system kernel.
125. **ISR -** Interrupt Service Routine/ Interrupt handler.
126. **Sti -** Set interrupt Flag.
127. **Cti -** Clear interrupt flag.
128. **Fork -** create a child process.
129. **Wait -** waits for a process to finish.
130. **MMX -** MultiMedia eXtension
131. **SSE -** Steaming SIMD Extensions
132. **AES-NI -** Advanced Encryption Standard New Instructions
133. **Dynamic Linking -** refers to the linking that is done during load or run-time and not when the exe is created.
134. **Static Linking -** is the result of the linker copying all library routines used in the program into the executable image.
135. **Shared Library -** is a file containing object code that several a. out files may use simultaneously while executing.

References

https://app.cybrary.it/browse/course/assembly