**Tugas 1  
Organisasi dan Arsitektur Komputer  
*“Instruction Set Processor X86 dan ARM”***

****

**NAMA : TYARA NOVITA WIDYANTI**

**NIM : L200150066**

**Program Studi Informatika  
Fakultas Komunikasi dan Informatika  
Universitas Muhammadiyah Surakarta**

**Set Instruksi dari x86**

|  |  |
| --- | --- |
| Opcode | Description |
| **AAA** | ASCII Adjust After Addition |
| **AAD** | ASCII Adjust AX Before Division |
| **AAS** | ASCII Adjust AL After Subtraction |
| **ADC** | Add with Carry |
| **ADD** | Add |
| **ADDPD** | Add Packed Double-Precision Floating-Point Values |
| **ADDPS** | Add Packed Single-Precision Floating-Point Values |
| **ADDSD** | Add Scalar Double-Precision Floating-Point Values |
| **ADDSS** | Add Scalar Single-Precision Floating-Point Values |
| **ADDSUBPD** | Packed Double-FP Add/Subtract |
| **ADDSUBPS** | Packed Single-FP Add/Subtract |
| **AND** | Logical AND |
| **ANDPD** | Bitwise Logical AND of Packed Double-Precision Floating-Point Values |
| **ANDPS** | Bitwise Logical AND of Packed Single-Precision Floating-Point Values |
| **ANDNPD** | Bitwise Logical AND NOT of Packed Double Precision Floating-Point Values |
| **ANDNPS** | Bitwise Logical AND NOT of Packed Single Precision Floating-Point Values |
| **ARPL** | Adjust RPL Field of Segment Selector |
| **BOUND** | Check Array Index Against Bounds |
| **BSF** | Bit Scan Forward |
| **BSR** | Bit Scan Reverse |
| **BSWAP** | Byte Swap |
| **BT** | Bit Test |
| **BTC** | Bit Test and Complement |
| **BTR** | Bit Test and Reset |
| **BTS** | Bit Test and Set |
| **CALL** | Call Procedure |
| **CBW/CWDE** | Convert Byte to Word/Convert Word to Doubleword |
| **CLC** | Clear Carry Flag |
| **CLD** | Clear Direction Flag |
| **CLFLUSH** | Flush Cache Line |
| **CLI** | Clear Interrupt Flag |
| **CLTS** | Clear Task-Switched Flag in CR0 |
| **CMC** | Complement Carry Flag |
| **CMOVcc** | Conditional Move |

|  |  |
| --- | --- |
| **CMP** | Compare Two Operands |
| **CMPPD** | Compare Packed Double-Precision Floating-Point Values |
| **CMPPS** | Compare Packed Single-Precision Floating-Point Values |
| **CMPS/CMPSB/CMP SW/CMPSD** | Compare String Operands |
| **CMPSD** | Compare Scalar Double-Precision Floating-Point Values |
| **CMPSS** | Compare Scalar Single-Precision Floating-Point Values |
| **CMPXCHG** | Compare and Exchange |
| **CMPXCHG8B** | Compare and Exchange 8 Bytes |
| **COMISD** | Compare Scalar Ordered Double-Precision Floating- Point Values and Set EFLAGS |
| **COMISS** | Compare Scalar Ordered Single-Precision Floating- Point Values and Set EFLAGS |
| **CPUID** | CPU Identification |
| **CVTDQ2PD** | Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values |
| **CVTDQ2PS** | Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point Values |
| **CVTPD2DQ** | Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTPD2PI** | Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTPD2PS** | Convert Packed Double-Precision Floating-Point Values to Packed Single-Precision Floating-Point Values |
| **CVTPI2PD** | Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values |
| **CVTPI2PS** | Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point Values |
| **CVTPS2DQ** | Convert Packed Single-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTPS2PD** | Convert Packed Single-Precision Floating-Point Values to Packed Double-Precision Floating-Point Values |
| **CVTPS2PI** | Convert Packed Single-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTSD2SI** | Convert Scalar Double-Precision Floating-Point Value to Doubleword Integer |
| **CVTSD2SS** | Convert Scalar Double-Precision Floating-Point Value to Scalar Single-Precision Floating-Point Value |

|  |  |
| --- | --- |
| **CVTSI2SD** | Convert Doubleword Integer to Scalar Double Precision Floating-Point Value |
| **CVTSI2SS** | Convert Doubleword Integer to Scalar Single Precision Floating-Point Value |
| **CVTSS2SD** | Convert Scalar Single-Precision Floating-Point Value to Scalar Double-Precision Floating-Point Value |
| **CVTSS2SI** | Convert Scalar Single-Precision Floating-Point Value to Doubleword Integer |
| **CVTTPD2PI** | Convert with Truncation Packed Double-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTTPD2DQ** | Convert with Truncation Packed Double-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTTPS2DQ** | Convert with Truncation Packed Single-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTTPS2PI** | Convert with Truncation Packed Single-Precision Floating-Point Values to Packed Doubleword Integers |
| **CVTTSD2SI** | Convert with Truncation Scalar Double-Precision Floating-Point Value to Signed Doubleword Integer |
| **CVTTSS2SI** | Convert with Truncation Scalar Single-Precision Floating-Point Value to Doubleword Integer |
| **CWD/CDQ** | Convert Word to Doubleword/Convert Doubleword to Quadword |
| **DAA** | Decimal Adjust AL after Addition |
| **DAS** | Decimal Adjust AL after Subtraction |
| **DEC** | Decrement by 1 |
| **DIV** | Unsigned Divide |
| **DIVPD** | Divide Packed Double-Precision Floating-Point Values |
| **DIVPS** | Divide Packed Single-Precision Floating-Point Values |
| **DIVSD** | Divide Scalar Double-Precision Floating-Point Values |
| **DIVSS** | Divide Scalar Single-Precision Floating-Point Values |
| **EMMS** | Empty MMX Technology State |
| **ENTER** | Make Stack Frame for Procedure Parameters |
| **F2XM1** | Compute 2x-1 |
| **FABS** | Absolute Value |
| **FADD/FADDP/FIADD** | Add |
| **FBLD** | Load Binary Coded Decimal |

|  |  |
| --- | --- |
| **FBSTP** | Store BCD Integer and Pop |
| **FCHS** | Change Sign |
| **FCLEX/FNCLEX** | Clear Exceptions |
| **FCMOVcc** | Floating-Point Conditional Move |
| **FCOM/FCOMP/FCO MPP** | Compare Floating Point Values |
| **FCOMI/FCOMIP/FUC OMI/FUCOMIP** | Compare Floating Point Values and Set EFLAGS |
| **FCOS** | Cosine |
| **FDECSTP** | Decrement Stack-Top Pointer |
| **FDIV/FDIVP/FIDIV** | Divide |
| **FDIVR/FDIVRP/FIDIV R** | Reverse Divide |
| **FFREE** | Free Floating-Point Register |
| **FICOM/FICOMP** | Compare Integer |
| **FILD** | Load Integer |
| **FINCSTP** | Increment Stack-Top Pointer |
| **FINIT/FNINIT** | Initialize Floating-Point Unit |
| **FIST/FISTP** | Store Integer |
| **FISTTP** | Store Integer with Truncation |
| **FLD** | Load Floating Point Value |
| **FLD1/FLDL2T/FLDL 2E/FLDPI/FLDLG2/F LDLN2/FLDZ** | Load Constant |
| **FLDCW** | Load x87 FPU Control Word |
| **FLDENV** | Load x87 FPU Environment |
| **FMUL/FMULP/FIMUL** | Multiply |
| **FNOP** | No operation |
| **FPATAN** | Partial Arctangent |
| **FPREM** | Partial Remainder |
| **FPREM1** | Partial Remainder |
| **FPTAN** | Partial Tangent |
| **FRNDINT** | Round to Integer |
| **FRSTOR** | Restore x87 FPU State |
| **FSAVE/FNSAVE** | Store x87 FPU State |
| **FSCALE** | Scale |
| **FSIN** | Sine |
| **FSINCOS** | Sine and Cosine |
| **FSQRT** | Square Root |
| **FST/FSTP** | Store Floating Point Value |
| **FSTCW/FNSTCW** | Store x87 FPU Control Word |
| **FSTENV/FNSTENV** | Store x87 FPU Environment |
| **FSTSW/FNSTSW** | Store x87 FPU Status Word |
| **FSUB/FSUBP/FISUB** | Subtract |

|  |  |
| --- | --- |
| **FSUBR/FSUBRP/FIS UBR** | Reverse Subtract |
| **FTST** | Test Floating Point Value |
| **FUCOM/FUCOMP/F UCOMPP** | Unordered Compare Floating Point Values |
| **FXAM** | Examine Floating Point Value |
| **FXCH** | Exchange Register Contents |
| **FXRSTOR** | Restore x87 FPU, MMX Technology, SSE, and SSE2 State |
| **FXSAVE** | Save x87 FPU, MMX Technology, SSE, and SSE2 State |
| **FXTRACT** | Extract Exponent and Mantissa |
| **FYL2X** | Compute y \* log\_2(x) |
| **FYL2XP1** | Compute y \* log\_2(x + 1) |
| **HADDPD** | Packed Double-FP Horizontal Add |
| **HADDPS** | Packed Single-FP Horizontal Add |
| **HLT** | Halt |
| **HSUBPD** | Packed Double-FP Horizontal Subtract |
| **HSUBPS** | Packed Single-FP Horizontal Subtract |
| **IDIV** | Signed Divide |
| **IMUL** | Signed Multiply |
| **IN** | Input from Port |
| **INC** | Increment by 1 |
| **INS/INSB/INSW/INS D** | Input from Port to String |
| **INT n/INTO/INT 3** | Call to Interrupt Procedure |
| **INVD** | Invalidate Internal Caches |
| **INVLPG** | Invalidate TLB Entry |
| **IRET/IRETD** | Interrupt Return |
| **Jcc** | Jump if Condition Is Met |
| **JMP** | Jump |
| **LAHF** | Load Status Flags into AH Register |
| **LAR** | Load Access Rights Byte |
| **LDDQU** | Load Unaligned Integer 128 Bits |
| **LDMXCSR** | Load MXCSR Register |
| **LDS/LES/LFS/LGS/L SS** | Load Far Pointer |
| **LEA** | Load Effective Address |
| **LEAVE** | High Level Procedure Exit |
| **LFENCE** | Load Fence |
| **LGDT/LIDT** | Load Global/Interrupt Descriptor Table Register |
| **LLDT** | Load Local Descriptor Table Register |
| **LMSW** | Load Machine Status Word |
| **LOCK** | Assert LOCK# Signal Prefix |

|  |  |
| --- | --- |
| **LODS/LODSB/LODS W/LODSD** | Load String |
| **LOOP/LOOPcc** | Loop According to ECX Counter |
| **LSL** | Load Segment Limit |
| **LTR** | Load Task Register |
| **MASKMOVDQU** | Store Selected Bytes of Double Quadword |
| **MASKMOVQ** | Store Selected Bytes of Quadword |
| **MAXPD** | Return Maximum Packed Double-Precision Floating- Point Values |
| **MAXPS** | Return Maximum Packed Single-Precision Floating-Point Values |
| **MAXSD** | Return Maximum Scalar Double-Precision Floating-Point Value |
| **MAXSS** | Return Maximum Scalar Single-Precision Floating Point Value |
| **MFENCE** | Memory Fence |
| **MINPD** | Return Minimum Packed Double-Precision Floating-Point Values |
| **MINPS** | Return Minimum Packed Single-Precision Floating-Point Values |
| **MINSD** | Return Minimum Scalar Double-Precision Floating Point Value |
| **MINSS** | Return Minimum Scalar Single-Precision Floating Point Value |
| **MONITOR** | Setup Monitor Address |
| **MOV** | Move |
| **MOV** | Move to/from Control Registers |
| **MOV** | Move to/from Debug Registers |
| **MOVAPD** | Move Aligned Packed Double-Precision Floating Point Values |
| **MOVAPS** | Move Aligned Packed Single-Precision Floating Point Values |
| **MOVD** | Move Doubleword |
| **MOVDDUP** | Move One Double-FP and Duplicate |
| **MOVDQA** | Move Aligned Double Quadword |
| **MOVDQU** | Move Unaligned Double Quadword |
| **MOVDQ2Q** | Move Quadword from XMM to MMX Technology Register |
| **MOVHLPS** | Move Packed Single-Precision Floating-Point Values High to Low |
| **MOVHPD** | Move High Packed Double-Precision Floating Point Value |
| **MOVHPS** | Move High Packed Single-Precision Floating-Point Values |

|  |  |
| --- | --- |
| **MOVLHPS** | Move Packed Single-Precision Floating-Point Values Low to High |
| **MOVLPD** | Move Low Packed Double-Precision Floating-Point Value |
| **MOVLPS** | Move Low Packed Single-Precision Floating-Point Values |
| **MOVMSKPD** | Extract Packed Double-Precision Floating-Point Sign Mask |
| **MOVMSKPS** | Extract Packed Single-Precision Floating-Point Sign Mask |
| **MOVNTDQ** | Store Double Quadword Using Non-Temporal Hint |
| **MOVNTI** | Store Doubleword Using Non-Temporal Hint |
| **MOVNTPD** | Store Packed Double-Precision Floating-Point Values Using Non-Temporal Hint |
| **MOVNTPS** | Store Packed Single-Precision Floating-Point Values Using Non-Temporal Hint |
| **MOVNTQ** | Store of Quadword Using Non-Temporal Hint |
| **MOVSHDUP** | Move Packed Single-FP High and Duplicate |
| **MOVSLDUP** | Move Packed Single-FP Low and Duplicate |
| **MOVQ** | Move Quadword |
| **MOVQ2DQ** | Move Quadword from MMX Technology to XMM Register |
| **MOVS/MOVSB/MOV SW/MOVSD** | Move Data from String to String |
| **MOVSD** | Move Scalar Double-Precision Floating-Point Value |
| **MOVSS** | Move Scalar Single-Precision Floating-Point Values |
| **MOVSX** | Move with Sign-Extension |
| **MOVUPD** | Move Unaligned Packed Double-Precision Floating- Point Values |
| **MOVUPS** | Move Unaligned Packed Single-Precision Floating Point Values |
| **MOVZX** | Move with Zero-Extend |
| **MUL** | Unsigned Multiply |
| **MULPD** | Multiply Packed Double-Precision Floating-Point Values |
| **MULPS** | Multiply Packed Single-Precision Floating-Point Values |
| **MULSD** | Multiply Scalar Double-Precision Floating-Point Values |
| **MULSS** | Multiply Scalar Single-Precision Floating-Point Values |
| **MWAIT** | Monitor Wait |
| **NEG** | Two's Complement Negation |

|  |  |
| --- | --- |
| **NOP** | No Operation |
| **NOT** | One's Complement Negation |
| **OR** | Logical Inclusive OR |
| **ORPD** | Bitwise Logical OR of Double-Precision Floating Point Values |
| **ORPS** | Bitwise Logical OR of Single-Precision Floating Point Values |
| **OUT** | Output to Port |
| **OUTS/OUTSB/OUTS W/OUTSD** | Output String to Port |
| **PACKSSWB/PACKS SDW** | Pack with Signed Saturation |
| **PACKUSWB** | Pack with Unsigned Saturation |
| **PADDB/PADDW/PA DDD** | Add Packed Integers |
| **PADDQ** | Add Packed Quadword Integers |
| **PADDSB/PADDSW** | Add Packed Signed Integers with Signed Saturation |
| **PADDUSB/PADDUS W** | Add Packed Unsigned Integers with Unsigned Saturation |
| **PAND** | Logical AND |
| **PANDN** | Logical AND NOT |
| **PAUSE** | Spin Loop Hint |
| **PAVGB/PAVGW** | Average Packed Integers |
| **PCMPEQB/PCMPEQ W/PCMPEQD** | Compare Packed Data for Equal |
| **PCMPGTB/PCMPGT W/PCMPGTD** | Compare Packed Signed Integers for Greater Than |
| **PEXTRW** | Extract Word |
| **PINSRW** | Insert Word |
| **PMADDWD** | Multiply and Add Packed Integers |
| **PMAXSW** | Maximum of Packed Signed Word Integers |
| **PMAXUB** | Maximum of Packed Unsigned Byte Integers |
| **PMINSW** | Minimum of Packed Signed Word Integers |
| **PMINUB** | Minimum of Packed Unsigned Byte Integers |
| **PMOVMSKB** | Move Byte Mask |
| **PMULHUW** | Multiply Packed Unsigned Integers and Store High Result |
| **PMULHW** | Multiply Packed Signed Integers and Store High Result |
| **PMULLW** | Multiply Packed Signed Integers and Store Low Result |
| **PMULUDQ** | Multiply Packed Unsigned Doubleword Integers |
| **POP** | Pop a Value from the Stack |
| **POPA/POPAD** | Pop All General-Purpose Registers |

|  |  |
| --- | --- |
| **POPF/POPFD** | Pop Stack into EFLAGS Register |
| **POR** | Bitwise Logical OR |
| **PREFETCHh** | Prefetch Data Into Caches |
| **PSADBW** | Compute Sum of Absolute Differences |
| **PSHUFD** | Shuffle Packed Doublewords |
| **PSHUFHW** | Shuffle Packed High Words |
| **PSHUFLW** | Shuffle Packed Low Words |
| **PSHUFW** | Shuffle Packed Words |
| **PSLLDQ** | Shift Double Quadword Left Logical |
| **PSLLW/PSLLD/PSL LQ** | Shift Packed Data Left Logical |
| **PSRAW/PSRAD** | Shift Packed Data Right Arithmetic |
| **PSRLDQ** | Shift Double Quadword Right Logical |
| **PSRLW/PSRLD/PSR LQ** | Shift Packed Data Right Logical |
| **PSUBB/PSUBW/PSU BD** | Subtract Packed Integers |
| **PSUBQ** | Subtract Packed Quadword Integers |
| **PSUBSB/PSUBSW** | Subtract Packed Signed Integers with Signed Saturation |
| **PSUBUSB/PSUBUS W** | Subtract Packed Unsigned Integers with Unsigned Saturation |
| **PUNPCKHBW/PUNP CKHWD/PUNPCKHD Q/PUNPCKHQDQ** | Unpack High Data |
| **PUNPCKLBW/PUNP CKLWD/PUNPCKLD Q/PUNPCKLQDQ** | Unpack Low Data |
| **PUSH** | Push Word or Doubleword Onto the Stack |
| **PUSHA/PUSHAD** | Push All General-Purpose Registers |
| **PUSHF/PUSHFD** | Push EFLAGS Register onto the Stack |
| **PXOR** | Logical Exclusive OR |
| **RCL/RCR/ROL/ROR** | Rotate |
| **RCPPS** | Compute Reciprocals of Packed Single-Precision Floating-Point Values |
| **RCPSS** | Compute Reciprocal of Scalar Single-Precision Floating- Point Values |
| **RDMSR** | Read from Model Specific Register |
| **RDPMC** | Read Performance-Monitoring Counters |
| **RDTSC** | Read Time-Stamp Counter |
| **REP/REPE/REPZ/RE PNE/REPNZ** | Repeat String Operation Prefix |
| **RET** | Return from Procedure |
| **RSM** | Resume from System Management Mode |

|  |  |
| --- | --- |
| **RSQRTPS** | Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values |
| **RSQRTSS** | Compute Reciprocal of Square Root of Scalar Single- Precision Floating-Point Value |
| **SAHF** | Store AH into Flags |
| **SAL/SAR/SHL/SHR** | Shift |
| **SBB** | Integer Subtraction with Borrow |
| **SCAS/SCASB/SCAS W/SCASD** | Scan String |
| **SETcc** | Set Byte on Condition |
| **SFENCE** | Store Fence |
| **SGDT** | Store Global Descriptor Table Register |
| **SHLD** | Double Precision Shift Left |
| **SHRD** | Double Precision Shift Right |
| **SHUFPD** | Shuffle Packed Double-Precision Floating-Point Values |
| **SHUFPS** | Shuffle Packed Single-Precision Floating-Point Values |
| **SIDT** | Store Interrupt Descriptor Table Register |
| **SLDT** | Store Local Descriptor Table Register |
| **SMSW** | Store Machine Status Word |
| **SQRTPD** | Compute Square Roots of Packed Double Precision Floating-Point Values |
| **SQRTPS** | Compute Square Roots of Packed Single Precision Floating-Point Values |
| **SQRTSD** | Compute Square Root of Scalar Double-Precision Floating-Point Value |
| **SQRTSS** | Compute Square Root of Scalar Single-Precision Floating-Point Value |
| **STC** | Set Carry Flag |
| **STD** | Set Direction Flag |
| **STI** | Set Interrupt Flag |
| **STMXCSR** | Store MXCSR Register State |
| **STOS/STOSB/STOS W/STOSD** | Store String |
| **STR** | Store Task Register |
| **SUB** | Subtract |
| **SUBPD** | Subtract Packed Double-Precision Floating-Point Values |
| **SUBPS** | Subtract Packed Single-Precision Floating-Point Values |
| **SUBSD** | Subtract Scalar Double-Precision Floating-Point Values |
| **SUBSS** | Subtract Scalar Single-Precision Floating-Point Values |

|  |  |
| --- | --- |
| **SYSENTER** | Fast System Call |
| **SYSEXIT** | Fast Return from Fast System Call |
| **TEST** | Logical Compare |
| **UCOMISD** | Unordered Compare Scalar Double-Precision Floating- Point Values and Set EFLAGS |
| **UCOMISS** | Unordered Compare Scalar Single-Precision Floating- Point Values and Set EFLAGS |
| **UD2** | Undefined Instruction |
| **UNPCKHPD** | Unpack and Interleave High Packed Double Precision Floating-Point Values |
| **UNPCKHPS** | Unpack and Interleave High Packed Single Precision Floating-Point Values |
| **UNPCKLPD** | Unpack and Interleave Low Packed Double Precision Floating-Point Values |
| **UNPCKLPS** | Unpack and Interleave Low Packed Single Precision Floating-Point Values |
| **VERR/VERW** | Verify a Segment for Reading or Writing |
| **WAIT/FWAIT** | Wait |
| **WBINVD** | Write Back and Invalidate Cache |
| **WRMSR** | Write to Model Specific Register |
| **XADD** | Exchange and Add |
| **XCHG** | Exchange Register/Memory with Register |
| **XLAT/XLATB** | Table Look-up Translation |
| **XOR** | Logical Exclusive OR |
| **XORPD** | Bitwise Logical XOR for Double-Precision Floating Point Values |
| **XORPS** | Bitwise Logical XOR for Single-Precision Floating Point Values |

**Set Instruksi dari ARM**

|  |  |
| --- | --- |
| **Instruction** | **Meaning** |
| **ABS** | Absolute Value |
| **ACS** | Arc Cosine |
| **ADC** | Add with Carry |
| **ADC** | Thumb: Add with Carry |
| **ADD** | Add |
| **ADD** | Thumb: Add |
| **ADF** | Add |
| **ADR** | Get address of object (within 4K) |
| **ADRL** | Get address of object (beyond 4K) |
| **ALIGN** | Set the program counter to the next word boundary |
| **AND** | Logical AND |
| **AND** | Thumb: Logical AND |
| **ASL** | Arithmetic Shift Left |
| **ASN** | Arc Sine |
| **ASR** | Arithmetic Shift Right |
| **ATN** | Arc Tangent |
| **B** | Branch |
| **B** | Thumb: Branch |
| **BIC** | Bit Clear |
| **BIC** | Thumb: Bit Clear |
| **BKPT** | Thumb: Breakpoint |
| **BL** | Branch with Link |
| **BL** | Thumb: Long Branch with Link |
| **BLX** | Thumb: Branch with Link and Exchange |
| **BX** | Thumb: Branch and Exchange |
| **CDP** | Co-processor data operation |
| **CDP2** | CDP, *non-conditional* so more co-processor commands possible |
| **CLZ** | Count Leading Zeros |
| **CMF** | Compare floating point value |
| **CMN** | Compare negated values |
| **CMN** | Thumb: Compare negated values |
| **CMP** | Compare values |
| **CMP** | Thumb: Compare values |
| **CNF** | Compare negated floating point values |
| **COS** | Cosine |
| **DCx** | Define byte (B), halfword (W), word (D), string (S), or floating point (F) value. Some assemblers allow DCFS, DCFD, etc for FP precision. |
| **DVF** | Divide |

|  |  |
| --- | --- |
| **EOR** | Exclusive-OR two values |
| **EOR** | Thumb: Logical Exclusive-OR |
| **EQUx** | Define byte (B), halfword (W), word (D), string (S), or floating point (F) value. Some assemblers allow EQUFS, EQUFD, etc for FP precision. |
| **EXP** | Exponent |
| **FABS** | VFP: Absolute |
| **FADD** | VFP: Addition |
| **FCMP** | VFP: Compare |
| **FCVTDS** | VFP: Single to Double |
| **FCVTSD** | VFP: Double to Single |
| **FCPY** | VFP: Copy [*like MVF*] |
| **FDIV** | VFP: Division |
| **FDV** | Fast Divide |
| **FIX** | Convert floating value to an integer |
| **FLD** | VFP: Load VFP registers |
| **FLDMDB** | VFP: Load multiple VFP registers, decr. before |
| **FLDMIA** | VFP: Load multiple VFP registers, incr. after |
| **FLT** | Convert integer to a floating value |
| **FMAC** | VFP: Multiply with Accumulate |
| **FMDHR** | VFP: Transfer ARM register to upper half of Double |
| **FMDLR** | VFP: Transfer ARM register to lower half of Double |
| **FMRDH** | VFP: Transfer upper half of Double to ARM register |
| **FMRDL** | VFP: Transfer lower half of Double to ARM register |
| **FML** | Fast multiply |
| **FMSC** | VFP: Multiply with Negate and Accumulate |
| **FMRS** | VFP: Transfer Single to ARM register |
| **FMSR** | VFP: Transfer ARM register to Single |
| **FMUL** | VFP: Multiply |
| **FMRX** | VFP: Transfer VFP system register to ARM register |
| **FMSTAT** | VFP: Transfer FPSCR flags to CPSR |
| **FMXR** | VFP: Transfer ARM register to VFP system register |
| **FNEG** | VFP: Copy Negative [*like MVN*] |
| **FNMAC** | VFP: Multiply with Deduct |
| **FNMSC** | VFP: Multiply with Negate and Deduct |
| **FNMUL** | VFP: Negative Multiply |
| **FRD** | Fast reverse divide |
| **FSITO** | VFP: Signed Integer to Float |
| **FSQRT** | VFP: Square Root |
| **FST** | VFP: Save VFP registers |
| **FSTMDB** | VFP: Save multiple VFP registers, decr. before |
| **FSTMIA** | VFP: Save multiple VFP registers, incr. after |
| **FSUB** | VFP: Subtraction |
| **FTOSI** | VFP: Float to Signed Integer |
| **FTOUI** | VFP: Float to Unsigned Integer |

|  |  |
| --- | --- |
| **FUITO** | VFP: Unsigned Integer to Float |
| **LDC** | Load from memory to co-processor |
| **LDC2** | LDC, *non-conditional* so more co-processor commands possible |
| **LDF** | Load floating point value |
| **LDM** | Load multiple registers |
| **LDMIA** | Thumb: Load multiple registers |
| **LDR** | Load register (32 bit) |
| **LDR** | Thumb: Load register (32 bits?) |
| **LDRB** | Load byte (8 bit) into register |
| **LDRB** | Thumb: Load byte (8 bit) into register |
| **LDRH** | Load halfword (16 bit) into register |
| **LDRH** | Thumb: Load halfwit (boo!) into register |
| **LDRSB** | Load signed byte (sign + 7 bit) into register |
| **LDRSB** | Thumb: Load signed byte (sign + 7 bit) into register |
| **LDRSH** | Load signed halfword (sign + 15 bit) into register |
| **LDRSH** | Thumb: Load signed halfword (sign + 15 bit) into register |
| **LFM** | Load multiple floating point values |
| **LGN** | Logarithm to base e |
| **LOG** | Logarithm to base 10 |
| **LSL** | Logical Shift Left |
| **LSR** | Logical Shift Right |
| **MCR** | Co-processor register transfer (ARM to co-processor) |
| **MCR2** | MCR, *non-conditional* so more co-processor commands possible |
| **MCRR** | MCR, with two registers transferred at one time |
| **MLA** | Multiply with Accumulate |
| **MNF** | Move negated |
| **MOV** | Move value/register into a register |
| **MOV** | Thumb: Move value/register into a register |
| **MRC** | Co-processor register transfer (co-processor to ARM) |
| **MRC2** | MRC, *non-conditional* so more co-processor commands possible |
| **MRRC** | MRC, with two registers transferred at one time |
| **MRS** | Move status flags to a register |
| **MSR** | Move contents of a register to the status flags |
| **MUF** | Multiply |
| **MUL** | Multiply |
| **MUL** | Thumb: Multiply |
| **MVF** | Move value/float register into a float register |
| **MVN** | Move negated |
| **MVN** | Thumb: Move negated |
| **NEG** | Thumb Negate |
| **NOP** | Thumb: No Operation |
| **NRM** | Normalise |
| **OPT** | Select assembly options |
| **ORR** | Logical OR |
| **ORR** | Thumb: Logical OR |

|  |  |
| --- | --- |
| **PLD** | PreLoaD |
| **POL** | Polar Angle |
| **POP** | Thumb: Pop registers from stack |
| **POW** | Power |
| **PUSH** | Thumb: Push registers onto stack |
| **QADD** | Add, saturating |
| **QDADD** | Add, double saturating |
| **QDSUB** | Subtract, double saturating |
| **QSUB** | Subtact, saturating |
| **RDF** | Reverse Divide |
| **RFC** | Read FP control register |
| **RFS** | Read FP status register |
| **RMF** | Remainder |
| **RND** | Round to integral value |
| **ROR** | Rotate Right |
| **RPW** | Reverse Power |
| **RRX** | Rotate Right with extend |
| **RSB** | Reverse Subtract |
| **RSC** | Reverse Subtract with Carry |
| **RSF** | Reverse Subtract |
| **SBC** | Subtract with Carry |
| **SBC** | Thumb: Subtract with Carry |
| **SFM** | Store Muliple Floating point values |
| **SIN** | Sine |
| **SMLA** | Signed Multiply with Accumulate of 16 bit \* 16 bit values |
| **SMLAL** | Signed Long (sign + 63 bit) Multiply with Accumulate |
| **SMLAL** | Signed Multiply with Accumulate of 16 bit \* 16 bit values, result is sign extended to 32 bits, then added to a 64 bit value. |
| **SMLAW** | Signed Multiply with Accumulate of 32 bit \* 16 bit values |
| **SMUL** | Signed Multiply of 16 bit \* 16 bit values |
| **SMULL** | Signed Long (sign + 63 bit) Multiply |
| **SMULW** | Signed Multiply of 32 bit \* 16 bit values |
| **SQT** | Square Root |
| **STC** | Co-processor data transfer |
| **STC2** | STC, *non-conditional* so more co-processor commands possible |
| **STF** | Store floating point value |
| **STM** | Store multiple registers |
| **STMIA** | Thumb: Store multiple registers |
| **STR** | Store a register (32 bit) |
| **STR** | Thumb: Store register (32 bit?) |
| **STRB** | Store a byte (8 bit) from a register |
| **STRB** | Thumb: Store byte (8 bit) |
| **STRH** | Store a halfword (16 bit) from a register |
| **STRH** | Thumb: Store halfword (16 bit) |
| **STRSB** | Store a signed byte (sign + 7 bit) from a register |

|  |  |
| --- | --- |
| **STRSH** | Store a signed half-word (sign + 15 bit) from a register |
| **SUB** | Subtract |
| **SUB** | Thumb: Subtract |
| **SUF** | Subtract |
| **SWI** | Cause a SoftWare Interrupt |
| **SWI** | Thumb: SoftWare Interrupt |
| **SWP** | Swap register with memory |
| **TAN** | Tangent |
| **TEQ** | Test Equivalence (notional EOR) |
| **TST** | Test bits (notional AND) |
| **TST** | Thumb: Test bits |
| **UMLAL** | Unsigned Long (64 bit) Multiply with Accumulate |
| **UMULL** | Unsigned Long (64 bit) Multiply |
| **URD** | Unnormalised round |
| **WFC** | Write FP control register |
| **WFS** | Write FP status register |