ARM Cortex-M são uma família de microcontroladores

Microcontroladores são diferentes de microprocessadores. Ao ponto que microprocessadores são apenas como uma CPU, e os Microcontroladores são os microprocessadores mais alguns periféricos(a placa de desenvolvimento toda em si)

Os microcontroladores funcionam com um sistema conjunto de registradores, que recebem informação a ser processada direto da memória e só salva na memória o que for preciso, os registradores podem receber desde o valor direto da variável à coisas mais especificas como o endereço na memória onde “começa” o espaço reservado para a RAM. Alguns registradores mais especiais recebem o endereço dos comandos(ex: R15(PC)), mostrando sempre qual próximo comando a ser executado

32 bits – 4 bytes  
16 bits – 2 bytes  
8 bits – 1 byte  
etc.

Big Endian -> byte mais significante no endereço mais “abaixo”

Little Endian -> byte menos significante no endereço mais “abaixo”

Microcontroladores Cortex-M usam Little Endian

Variáveis globais são armazenadas direto na RAM

Set de instruções ARM: 32 bits, máxima perfomance, menor densidade de código  
Set de instruções Thumb 1: 16 bits, mínima performance, maior densidade de código  
Set de instruções Thumb 2: Mistura 16 bits e 32 bits, para maior performance sem perder a densidade de código

A família Cortex-M tem compatibilidade upward, o que roda num M0, M0+ e M1, roda num M3, no M4 e no M7. Sempre sucessivo para cima, nunca para baixo

Na sintaxe da linguagem assembly temos quatro campos separados por espaços ou tabs: Label, Opcode(operation code), Operand e Comment. No caso, labels e comentários são campos opcionais, mas constituem parte das boas práticas na programação

Label: Usado para encontrar a posição da intrução atual na memória  
Opcode: Especifica a operação para o processador performar  
Operand: Especifica uma destinação/source para que os dados sejam processados pelo opcode(pode ser um número, um registrador, etc).  
Comment: Usado para explicar o significado do código

Directives: não são instruções propriamente ditas mas performam certas funções similares à instruções. Eles ajudam e controlam processos assembly. Às vezes chamados de pseudo-ops. Não fazem parte do set de instruções. Eles mudam a maneira de como o código é assimilado

Tipos de Directives:   
Thumb: “Colocado” no topo do arquivo para especificar que o código é gerado pelo set de de instruções Thumb  
CODE: Denota a sessão para instruções de máquina (ROM)  
DATA: Denota a sessão para variáveis globais (RAM)  
AREA: Instrui o “assimilador” a assimilar um novo código ou sessão de dados  
SPACE: Reserva um bloco de memória e preenche ele com “zeros”  
ALIGN: Usado para grantir que o próximo objeto está devidamente alinhado  
EXPORT: Para fazer um objeto acessível a partir de outro arquivo  
GLOBAL: Mesmo significado de EXPORT  
IMPORT: Acessar um objeto “exportado”  
END: Colocado no fim de cada arquivo  
DCB: Põe(declara) um byte(8-bits) constante na memória  
DCW: Põe(declara) uma “meia-palavra”(16-bits) constante na memória  
DCD: Põe(declara) uma palavra(32-bits) constante na memória  
EQU: Para dar um nome simbólico para uma constante numérica  
.|text|: Faz com que o código de montagem seja “chamável” em C

Existem 6 “Operações Bitwise:

1. AND: &
2. OR: |
3. EOR: ^
4. NOT: ~
5. RIGHT SHIFT: >>
6. LEFT SHIFT: <<

Essas operações servem em si para facilitar o acesso aos endereços de memória e na maior parte das vezes faz com que mudemos apenas os bits necessários.

CMSIS = Cortex Microcontroller Software Interface Standart