

---

---

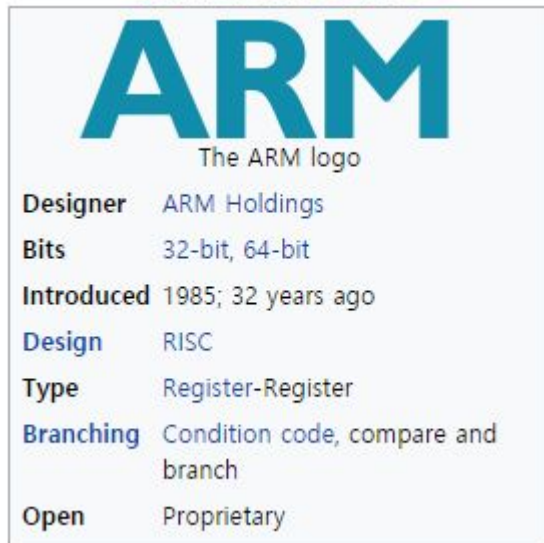
# ARM 프로세서 개요

Hancheol Cho

---

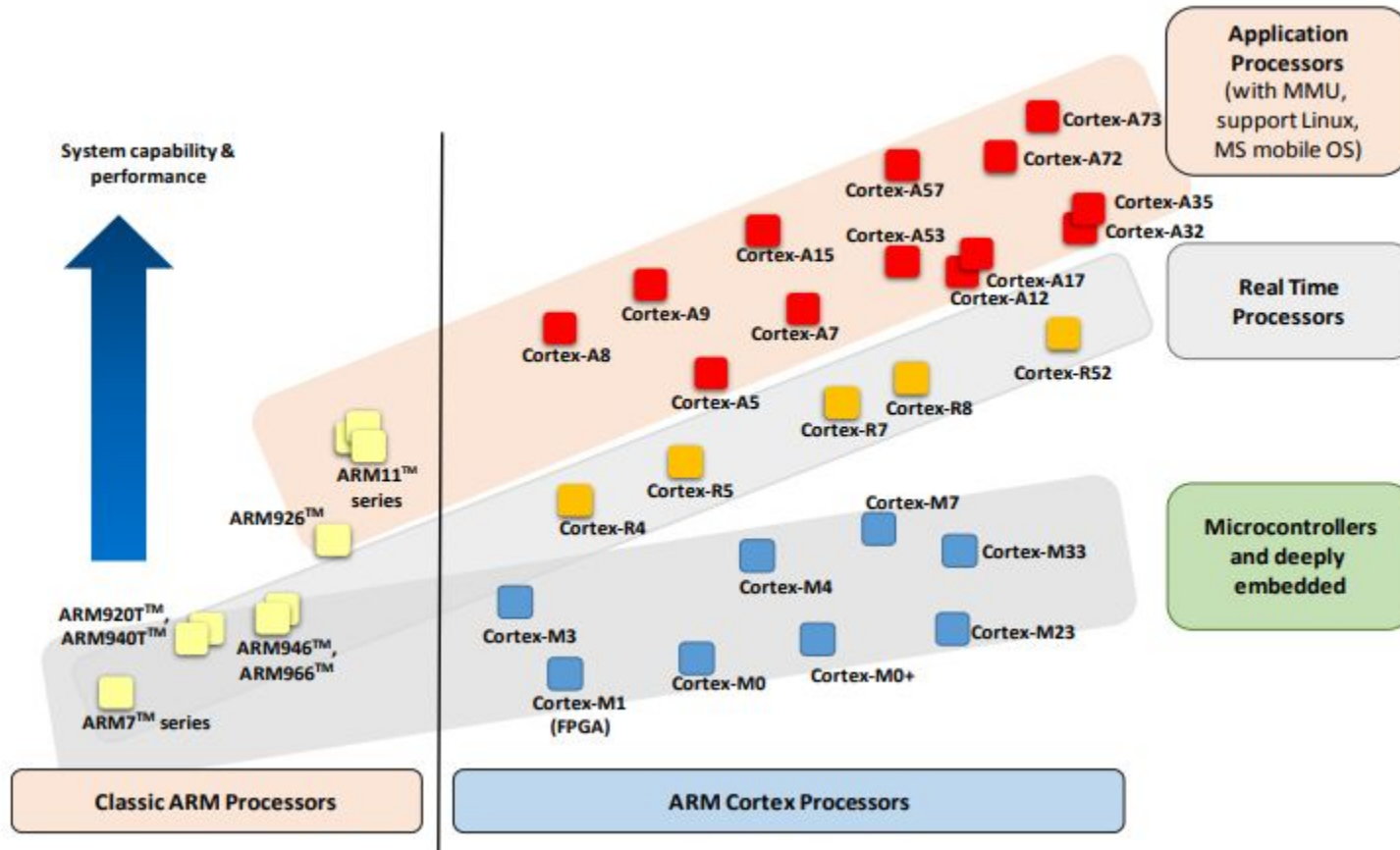
---

# ARM (Advanced RISC Machine)



**ARM**, originally **Acorn RISC Machine**, later **Advanced RISC Machine**, is a family of reduced instruction set computing (RISC) architectures for computer processors, configured for various environments. **British company ARM Holdings develops the architecture and licenses it to other companies**, who design their own products that implement one of those architectures

# ARM Processor Family



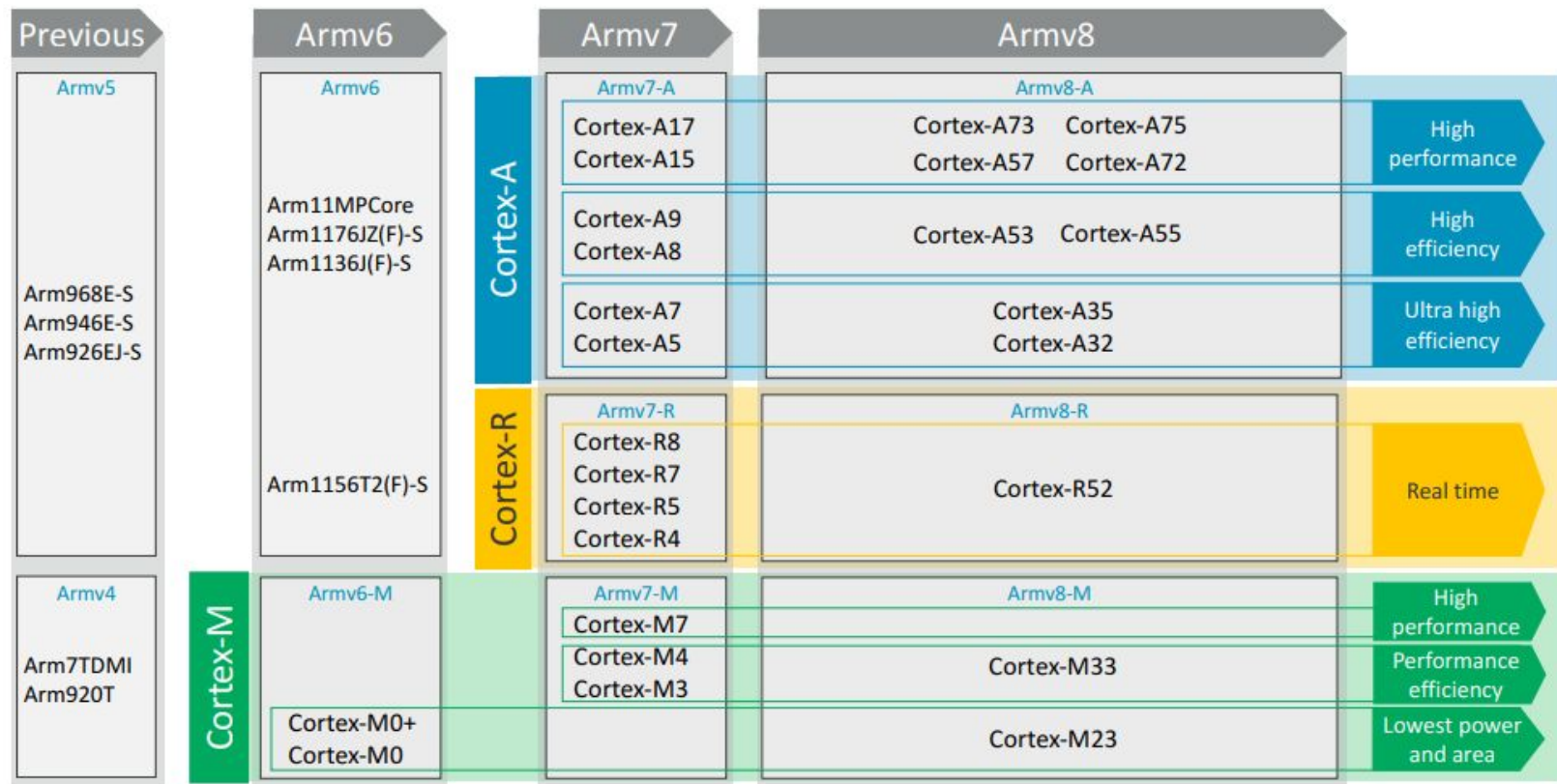
출처 : arm.com

# ARM Architecture 분류

- A** The *Application* profile defines a VMSA based microprocessor architecture. It is targeted at high performance processors, capable of running full feature operating systems. It supports the ARM and Thumb instruction sets.
- R** The *Real-time* profile defines a PMSA based microprocessor architecture. It is targeted at systems that require deterministic timing and low interrupt latency. It supports the ARM and Thumb instruction sets.
- M** The *Microcontroller* profile provides low-latency interrupt processing accessible directly from high-level programming languages. It has a different exception handling model to the other profiles, implements a variant of the PMSA, and supports a variant of the Thumb instruction set only.

Profile	Architecture	Instruction Set	Processor
A-Profile	ARMv7-A	A32, T32	Cortex-A Series
R-Profile	ARMv7-R	A32, T32	Cortex-R Series
M-Profile	ARMv7-M	T32	Cortex-M Series
	ARMV6-M	T32	Cortex-M0 Series

# ARM Processor

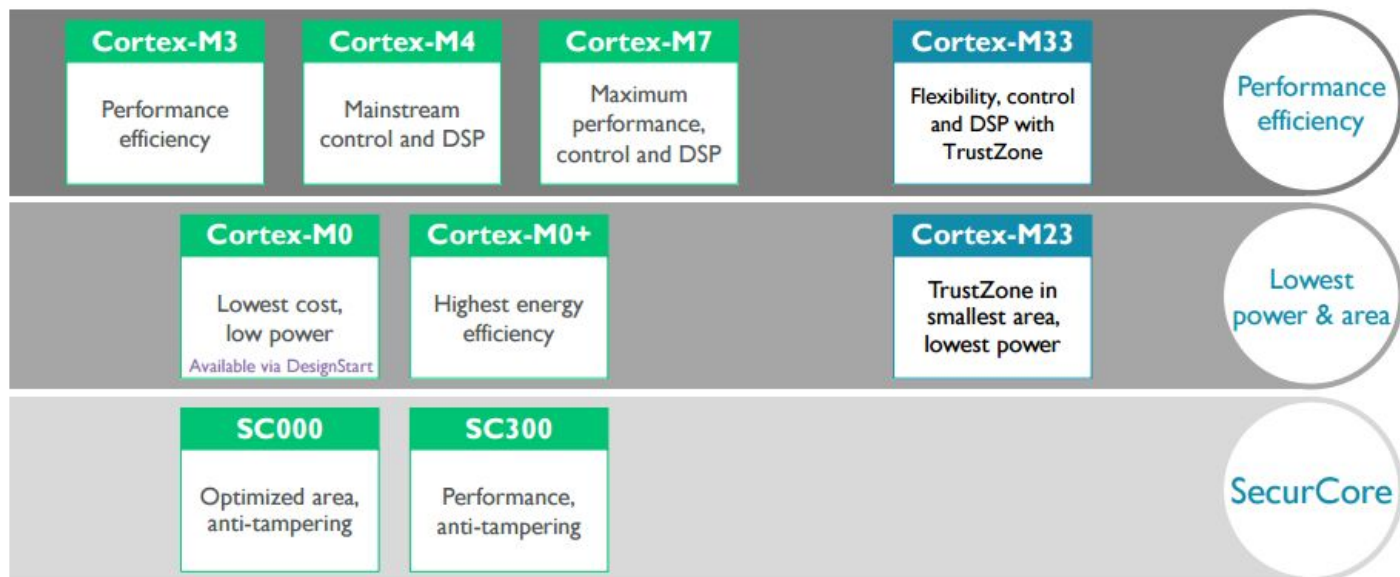


© 2017 Arm Limited

arm

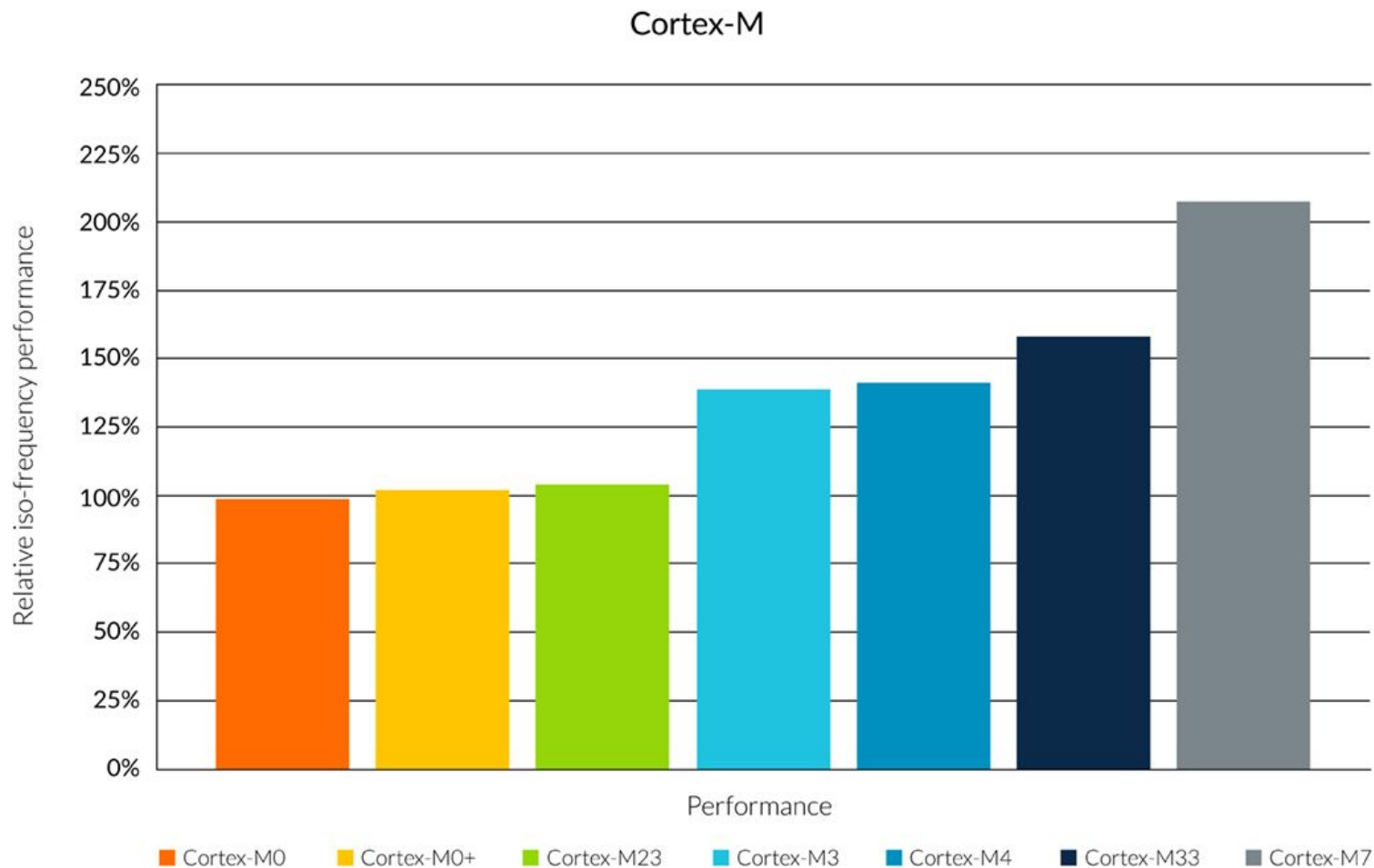
출처 : arm.com

# Cortex-M Processor



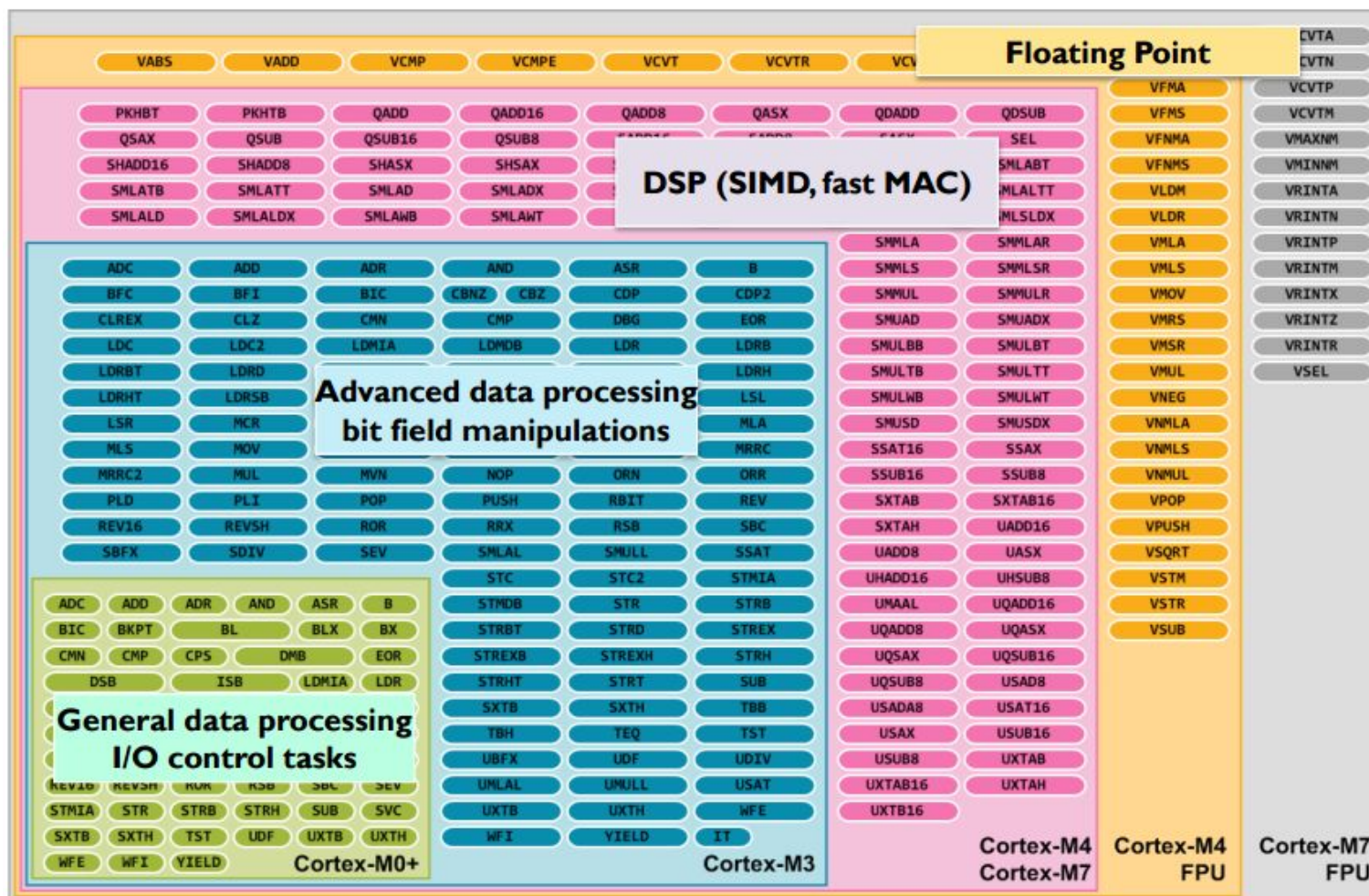
	Cortex-M0	Cortex-M0+	Cortex-M3	Cortex-M4	Cortex-M7
Instruction set	ARMv6-M	ARMv6-M	ARMv7-M	ARMv7-M	ARMv7-M
architecture	Thumb, Thumb-2	Thumb, Thumb-2	Thumb, Thumb-2	Thumb, Thumb-2, DSP, FP (SP)	Thumb, Thumb-2, DSP, FP (1. SP or 2. SP+DP)

# Cortex-M Performance



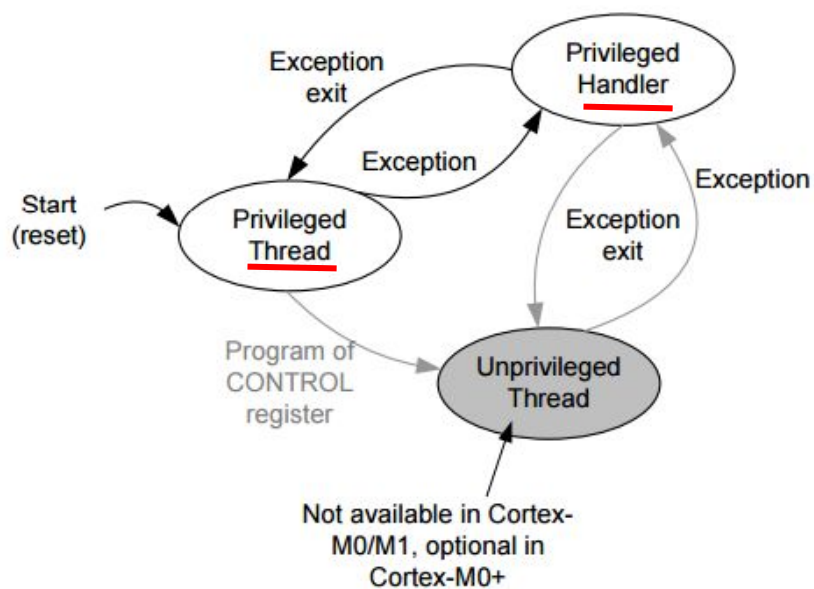


# Cortex-M Instruction Set support





# Programmer's model



General registers

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12

R13 (MSP)	Main Stack Pointer (MSP), Process Stack Pointer (PSP)
R13 (PSP)	
R14	Link Register (LR)
R15	Program Counter (PC)

Available on the Cortex-M4 with FPU only

Floating Point Unit

S1	S0	D0
S3	S2	D1
S5	S4	D2
S7	S6	D3
S9	S8	D4
S11	S10	D5
S13	S12	D6
S15	S14	D7
S17	S16	D8
S19	S18	D9
S21	S20	D10
S23	S22	D11
S25	S24	D12
S27	S26	D13
S29	S28	D14
S31	S30	D15

FPSCR Floating Point Status and Control Register

Not available in ARMv6-M

Name	Functions	
xPSR	Program Status Registers	Special Registers
PRIMASK		
FAULTMASK	Interrupt Mask Registers	
BASEPRI		
CONTROL	Control Register	

**Real Time OS ?**

# Context

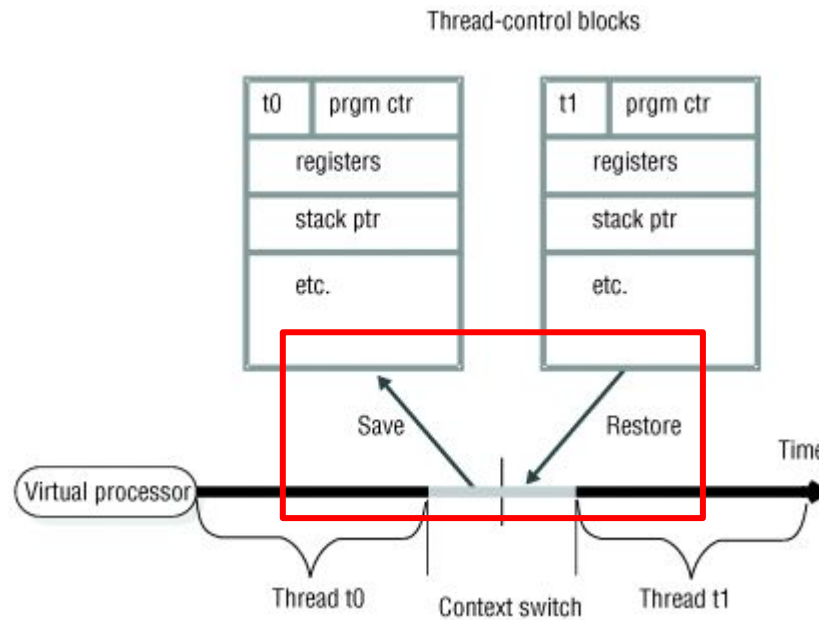
- 스레드/태스크가 실행되기 위한 최소한의 데이터
  - CPU 레지스터, 스택포인터 등등..

System & User	FIQ	Supervisor	Abort	IRQ	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

*ARM State Program Status Registers*

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

# Context Switching



출처

# 스케줄러

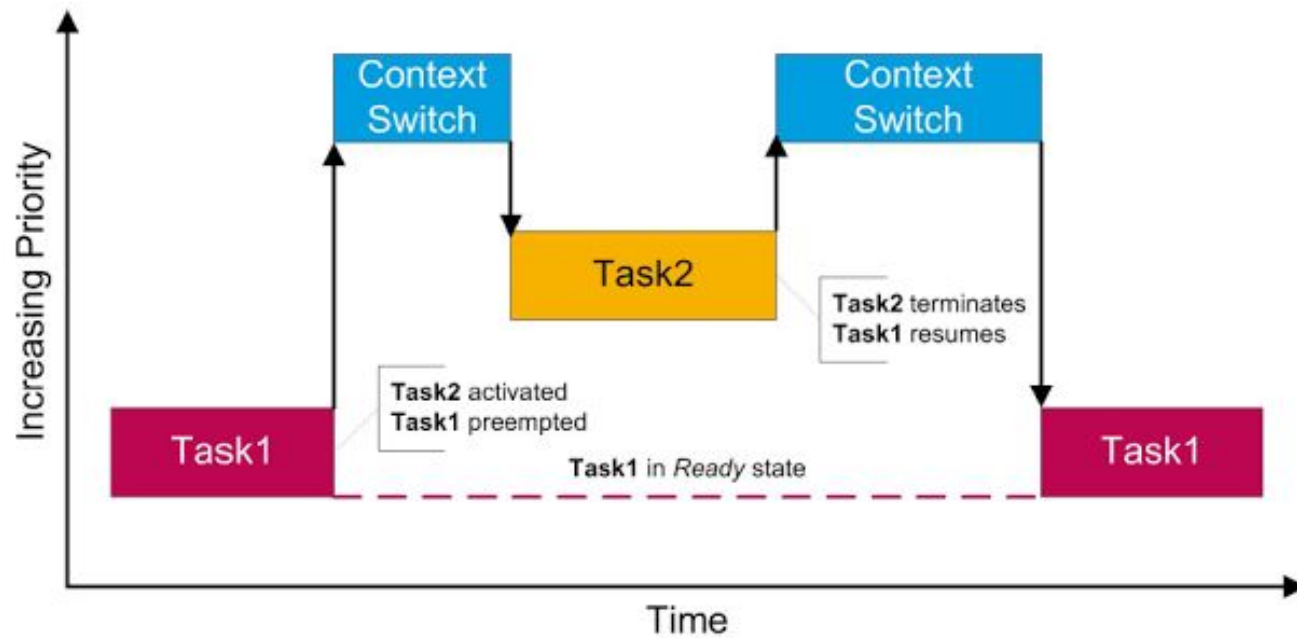
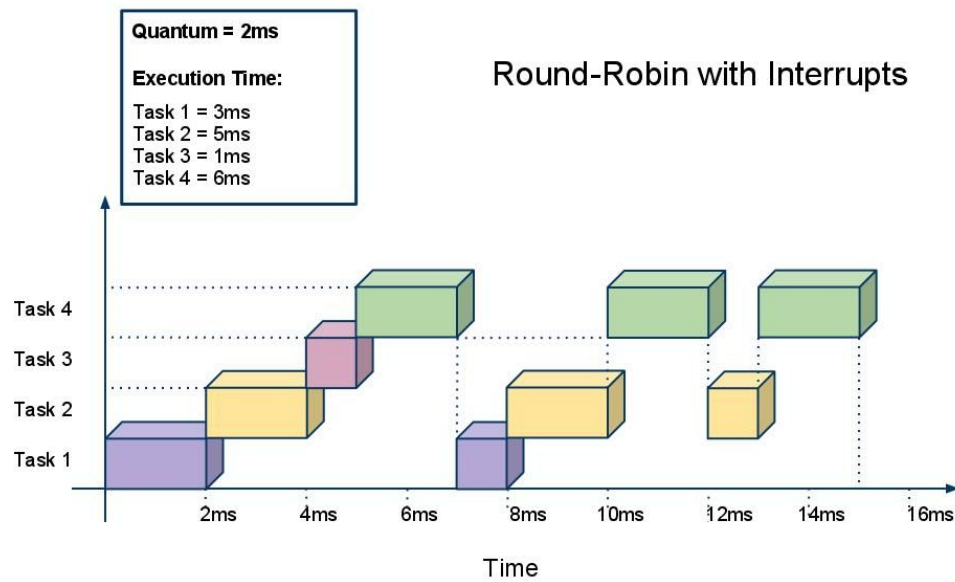


Figure 4.1: Preemptive scheduling of tasks

# 스케줄러





# RTOS



# Exception Vector

- 스택포인터 초기값을 지정 가능
  - 스타트업 코드도 C언어로 작성 가능

Exception Type	ARMv6-M	ARMv7-M	Vector Table	Vector address (initial)
255			Interrupt#239 vector	0x000003FC
47		Device Specific Interrupts	Interrupt#31 vector	0x000000BC
17	Device Specific Interrupts		Interrupt#1 vector	0x00000044
16			Interrupt#0 vector	0x00000040
15	SysTick	SysTick	SysTick vector	0x0000003C
14	PendSV	PendSV	PendSV vector	0x00000038
13	Not used	Not used	Not used	0x00000034
12		Debug Monitor	Debug Monitor vector	0x00000030
11	SVC	SVC	SVC vector	0x0000002C
10			Not used	0x00000028
9			Not used	0x00000024
8		Not used	Not used	0x00000020
7	Not used		SecureFault (ARMv8-M Mainline)	0x0000001C
6		Usage Fault	Usage Fault vector	0x00000018
5		Bus Fault	Bus Fault vector	0x00000014
4		MemManage (fault)	MemManage vector	0x00000010
3	HardFault	HardFault	HardFault vector	0x0000000C
2	NMI	NMI	NMI vector	0x00000008
1			Reset vector	0x00000004
0			MSP initial value	0x00000000

# 성능 비교

- 데이터 처리 속도

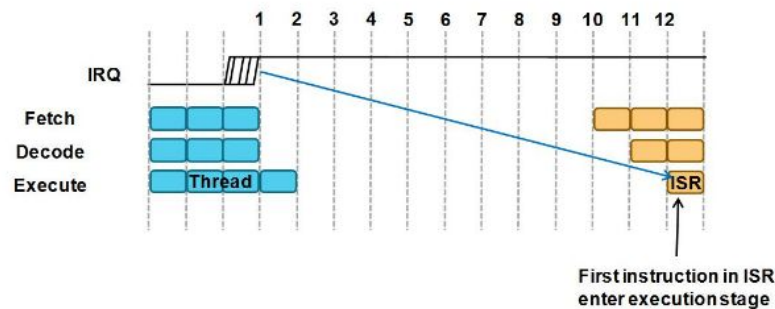
	Dhrystone DMIPS/MHz (v2.1) – official	Dhrystone DMIPS/MHz (v2.1) – full optimization	Coremark/MHz (v1.0)
Cortex-M0	0.84	1.21	2.33
Cortex-M0+	0.94	1.31	2.42
Cortex-M3	1.25	1.89	3.32
Cortex-M4	1.25	1.95	3.40
Cortex-M7	2.14	2.55	5.01
Cortex-M23	0.98	-	2.5
Cortex-M33	1.5	-	3.86

- Interrupt Latency

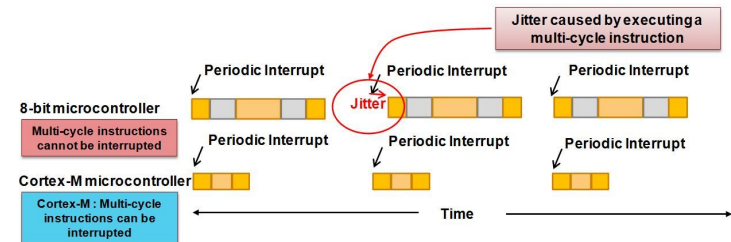
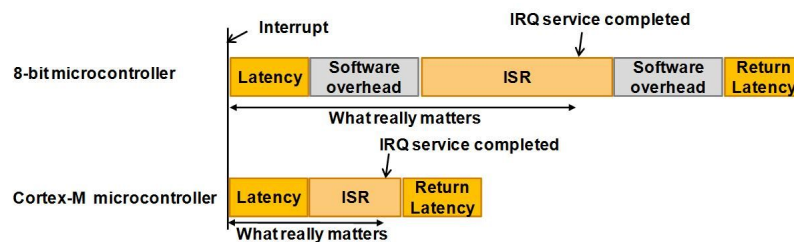
	Interrupt latency (number of clock cycles)
Cortex-M0	16
Cortex-M0+	15
Cortex-M23	15
Cortex-M3	12
Cortex-M4	12
Cortex-M7	Typically 12, worst case 14
Cortex-M33	12

# Interrupt Latency

- Interrupt Latency ?



- 기존 마이크로 컨트롤러와 비교



## 7.1 Why Cortex-M processors are easy to use

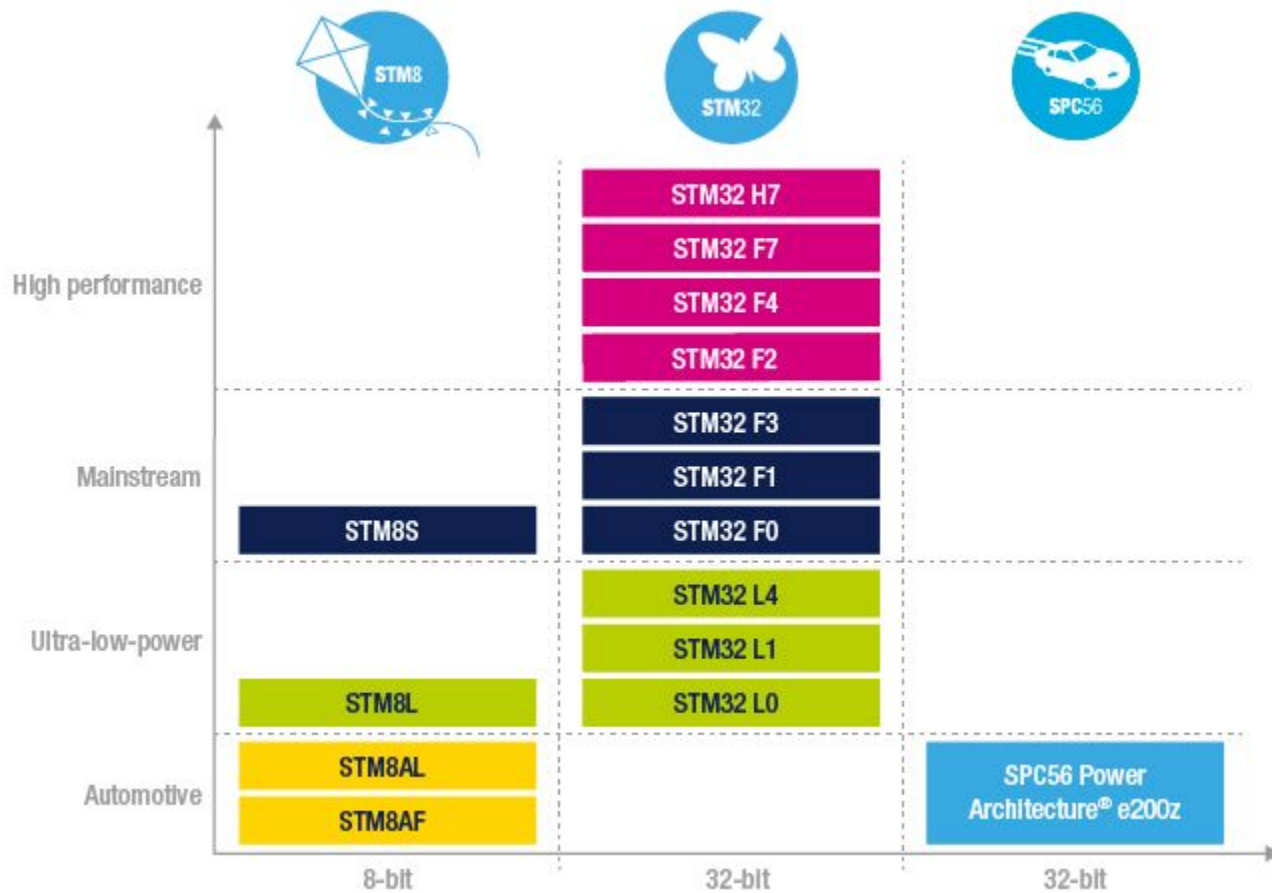
Although the Cortex-M processors are packed with features, they are also very easy to use. For instance, almost everything can be programmed in high-level language like C. Although there is a big variety of different Cortex-M processor-based products (e.g. with different memory size, peripherals, performance, packages, etc), the consistency of the architecture make it easy to start using a new Cortex-M processor once you have experience with one of them.

To make software development easier, and to enable better software reusability and portability, ARM developed the CMSIS-CORE, where CMSIS stands for Cortex Microcontroller Software Interface Standard. The CMSIS-CORE provides a standardized Hardware Abstraction Layer (HAL) for various features in the processors such as interrupt management control, using a set of APIs. The CMSIS-CORE is integrated in the device driver libraries from various microcontroller vendors, and is supported by various compilation suites.

Beside from CMSIS-CORE, CMSIS also have a DSP software library (CMSIS-DSP). It provides various DSP functions and is optimized for Cortex-M4 and Cortex-M7 processors, and also supports other Cortex-M processors. Both CMSIS-CORE and CMSIS-DSP are free and can be downloaded from the GitHub ([CMSIS 4](#), [CMSIS 5](#)), and are supported by multiple tool vendors.

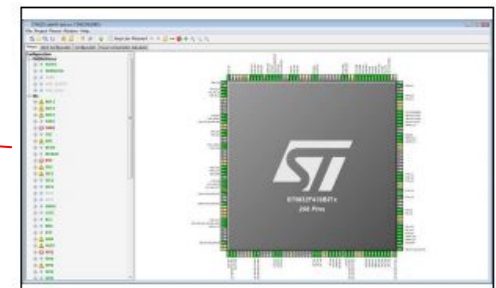
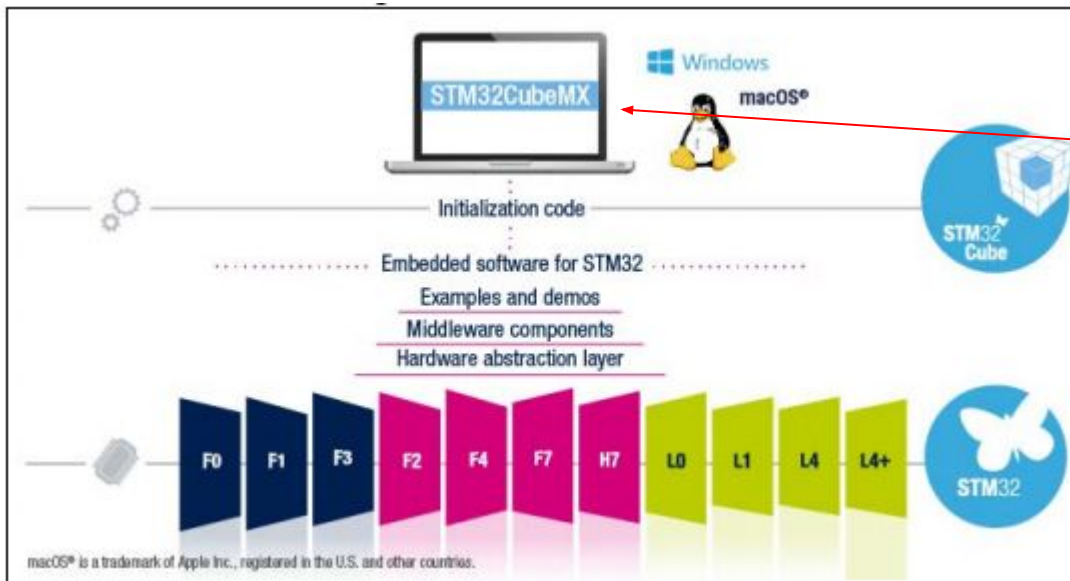


# ST사의 MCU 구성

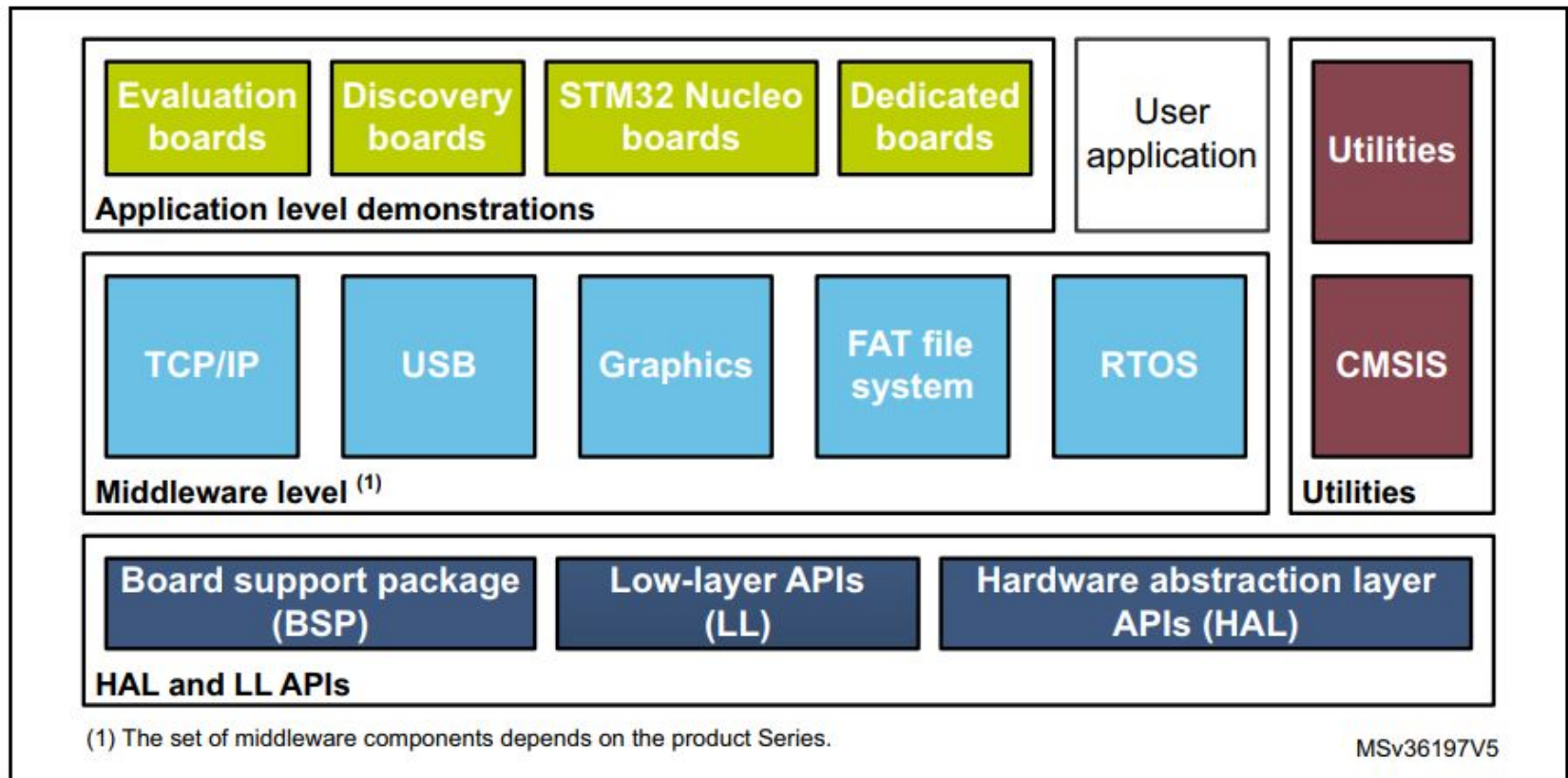




# STM32CubeMX, STM32Cube

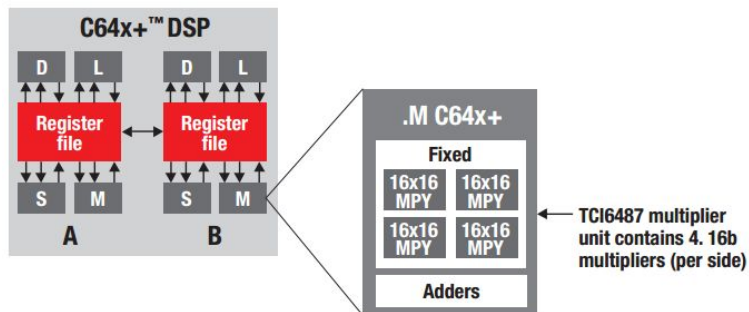


# STM32Cube



# DSP ?

- DSP 장점
  - 연산모듈이 8개 (최대 1Clock에 8개의 명령어 실행 가능)
  - Very-Long-Instruction-Word (VLIW)
  - 데이터 버스 최대 256bit



- 소프트웨어 파이프라인을 통한 병렬 실행

```
MVKL L1DCC, A0 ; \
|| MVKL L1PCC, B0 ; |__ Generate L1DCC pointer in A0
MVKH L1DCC, A0 ; |__ and L1PCC pointer in B0
|| MVKH L1PCC, B0 ; /
|| MVK 1b, A1 ; \__ OPER encoding for 'freeze'
|| MVK 1b, B1 ; /__ in both A1 and B1.
STW A1, *A0 ; Write to L1DCC.OPER
|| STW B1, *B0 ; Write to L1PCC.OPER
LDW *A0, A1 ; Get old freeze state into A1 from L1DCC
|| LDW *B0, B1 ; Get old freeze state into B1 from L1PCC
NOP 4
; At this point, L1D and L1P are frozen.
; The old value of L1DCC.OPER is in bit 16 of A1.
; The old value of L1PCC.OPER is in bit 16 of B1.
```

# DSP ?

- DSP 장점

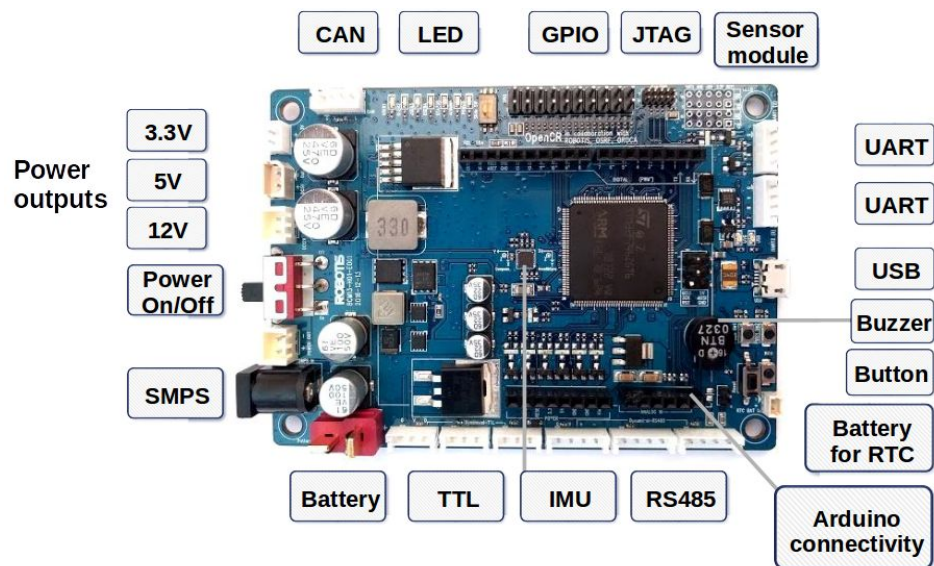
- DMA 기능이 강력함
  - DMA 기능만으로도 일부 이미지 처리가 가능함

- DSP 단점

- 인터럽트 발생시 명령어가 길어서 연산속도에 영향을 많음
  - 최적화시 기본적으로 인터럽트가 **Disable**됨으로 인터럽트 사용시에는 최적화 옵션 사용시 주의가 필요함
  - ARM 프로세서와 듀얼로 많이 사용
- 최적화에 따른 속도 편차가 심함
  - 연산모듈은 8개이나 명령어 종류에 따른 동시 실행이 안되는 경우가 있음
  - 컴파일러 옵션만으로는 최적화의 한계가 있으므로 TI에서 제공하는 최적화 라이브러리 사용 권장
- 캐시에 대한 영향이 크다
  - 명령어도 길고 데이터도 크기때문에 캐시 메모리에서 실행시와 외부메모리에서 실행시 속도 편차가 큼

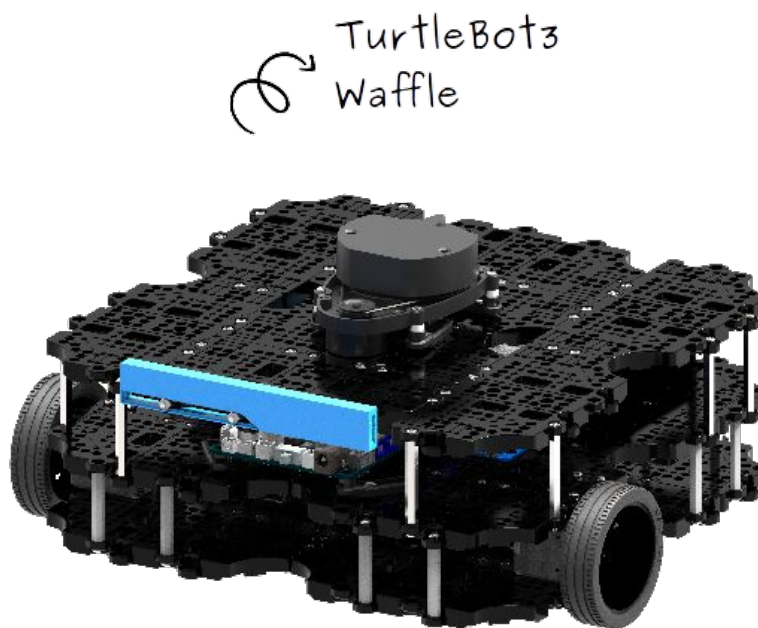
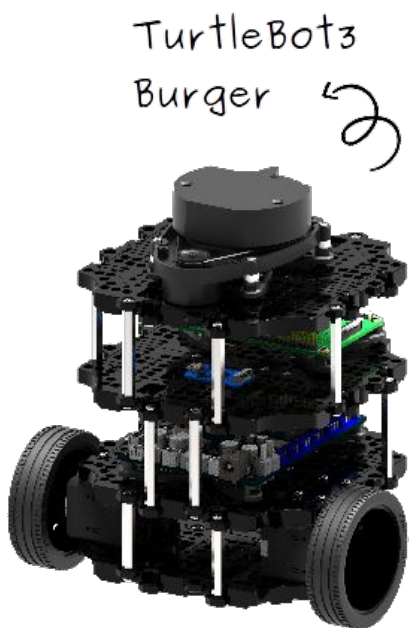
# OpenCR (Open-source Control Module for ROS)

- STM32F746ZGT6 216Mhz, Cortex-M7, 1MB Flash, 320KB SRAM
- 아두이노 우노 핀 헤더
- 아두이노 IDE 개발환경 지원
- 다이내믹셀/올로/UART/CAN 인터페이스
- 배터리 입력 및 전원 출력(12V/5V/3.3V)



# OpenCR (Open-source Control Module for ROS)

- Turtlebot3 Burger/Waffle의 제어기로 사용됨





# OpenCR (Open-source Control Module for ROS)

- 활용 예제

<https://youtu.be/-kBfIS6wJs>



# OpenCR (Open-source Control Module for ROS)

- 하드웨어 자료
  - <https://github.com/ROBOTIS-GIT/OpenCR-Hardware>
- 펌웨어 자료
  - <https://github.com/ROBOTIS-GIT/OpenCR>