PROCESSORS ARCHITECTURE OVERVIEW

ARM & AVR

Hardvardská architektura = Data jinde než program Procesor ARM obsahuje 44 základních instrukcí s jednotnou šířkou 32 bitů. V jednom taktu se vykonávají pouze instrukce pracující s aritmeticko-logickou jednotkou (ALU), s registry nebo s přímými operandy

2017@Václav Novák

http://www.firmcodes.com/microcontrollers/arm/arm-processors-architecture-overview/

SÉRIE ARM

Série ARM Classic

Klasická řada ARM se týká procesorů začínajících od ARM7 po ARM11. Jedná se o sérii, která dává trhový impuls ARM díky svým základním funkcím jako Data Tightly Coupled memory, cache, MMU, MPU atd. Typickými příklady této série jsou ARM7TDMI, ARM926EJ-S, ARM11 MPCore atd.

Série Cortex-A: Profil A, profil "Aplikace"

V této architektuře Cortex přenášíme různé embedded OS a design embedded system systémem OS. Jádrem tohoto profilu je nejvyšší výkon s nízkým výkonem, TrustZone a Jazelle-RCT pro bezpečný a rozšiřitelný systém. Praktická platforma pro vývoj těchto typů profilů je FriendlyARM, Malina Pi, atd.

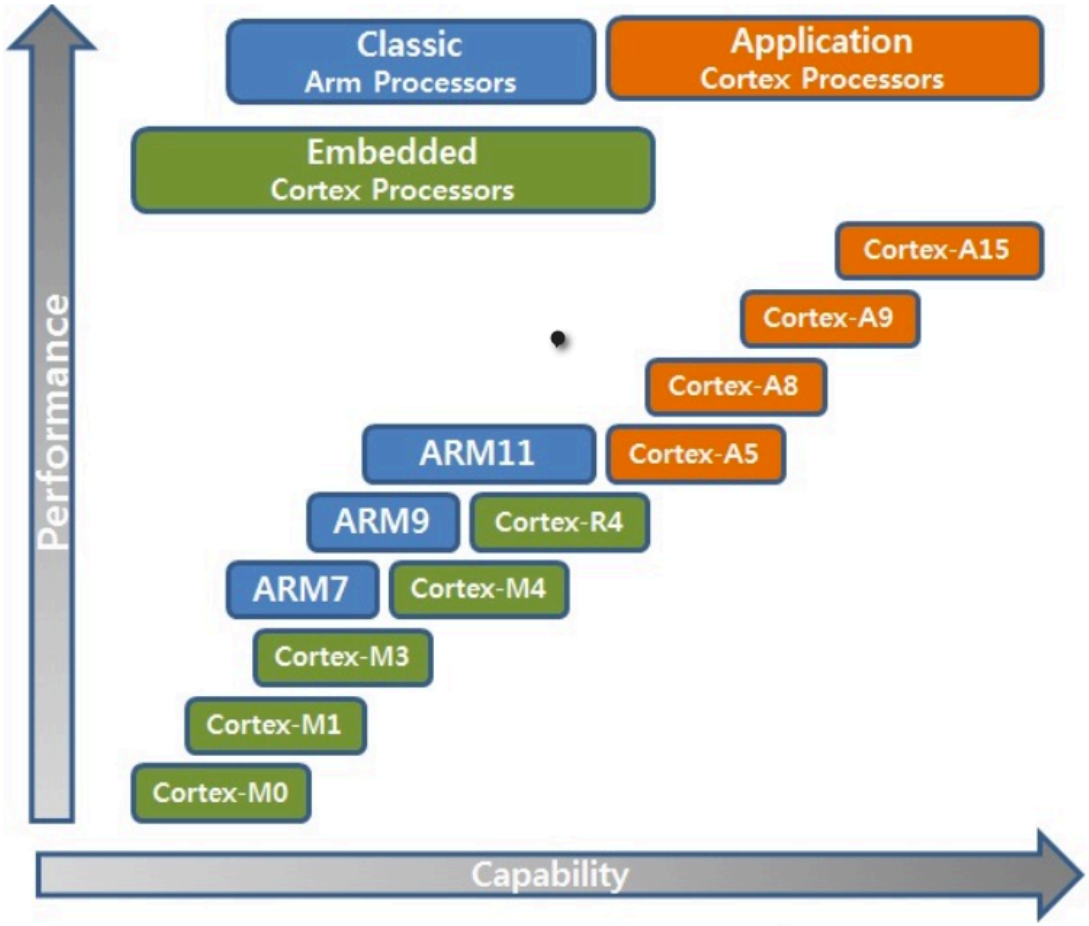
Série Cortex-R: profil R, profil "v reálném čase":

Jedná se o Cortex architekturu, která se většinou používá v reálném čase, kde aplikace přeruší kritickou situaci. Obsahuje základní funkce, jako jsou potřeby chráněné paměti (MPU), nízká latence a předvídatelnost v reálném čase.

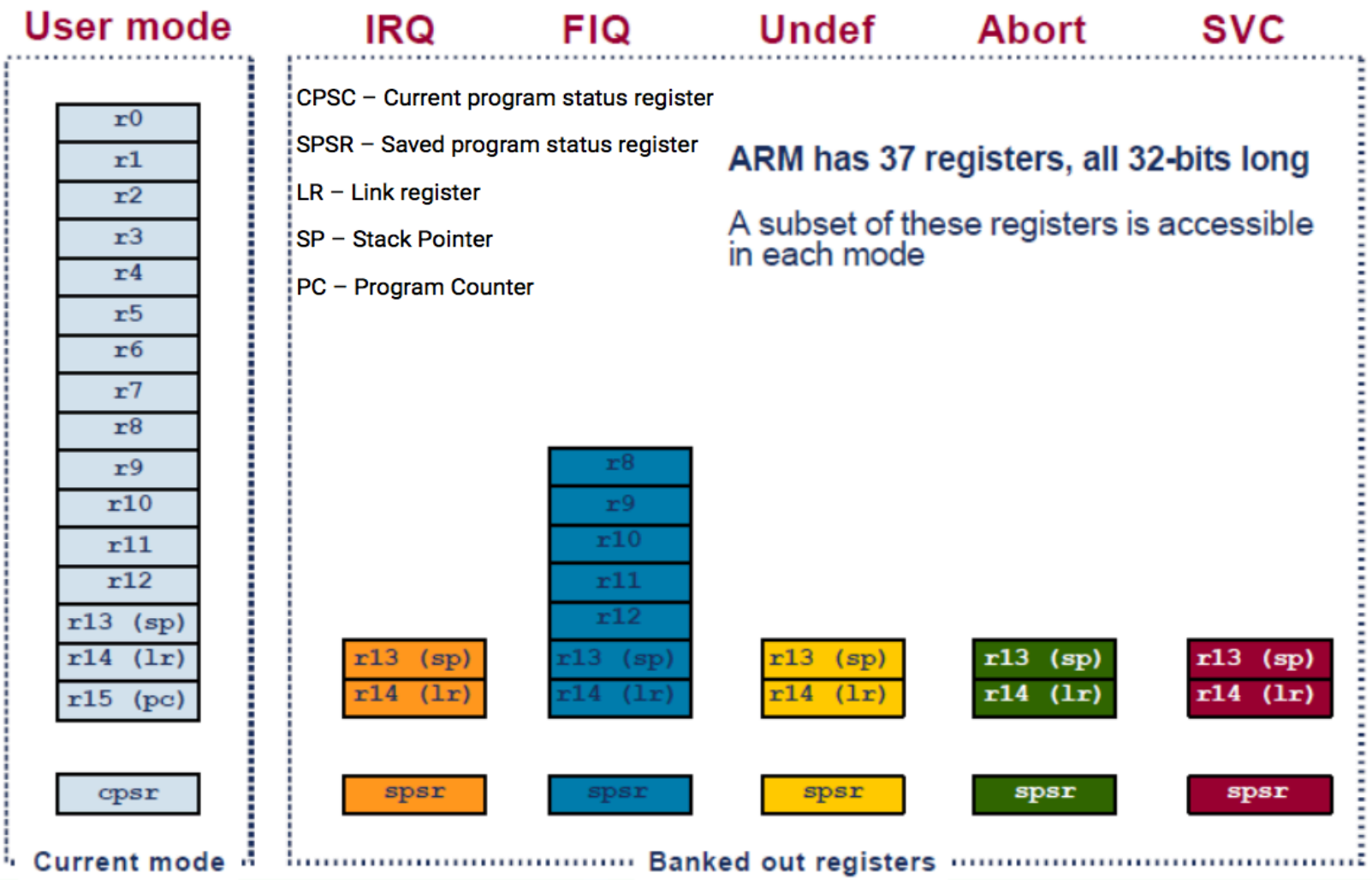
Série Cortex-M: profil M, profil "mikrokontroléru":

Tento profil je speciálně určen pouze pro účely mikrokontroléru. Klíčová vlastnost tohoto profilu je jako vstupní bod Nejnižší brána, Deterministické a předvídatelné chování, klíčová priorita, Hluboce zakořeněné použití. Typickým příkladem této profilové architektury je Hercules TMS470M, STM32F429 atd.

ARM ARCHITECTURE



ARM REGISTRY



MICROCHIP Atmel

AVR

Standard AVR pro Alf Vegard RISC je modifikovaná architektura Harvard, 8-bitová RISC, jednočipový mikrokontrolér navržený dvěma studenty norského institutu technologie, Alf-Egil Bogen a Vegard wollan a poté byl vyvinut Atmel v roce 1996. Společnost AVR byla jednou z prvních rodin mikrokontrolérů, která používala flash paměť na paměťovou kartu pro programové úložiště, na rozdíl od OTP ROM, EPROM nebo EEPROM, které v té době používaly jiné mikroprocesory.

MICROCHIP Atmel

AVR

- **CLASSIC**
  - AT9052313
  - AT9054433
  - AT89c51
- **MEGA**
  - ATmega8
  - ATmega16
  - ATmega32
  - ATmega128
- **TINY**
  - ATtiny13
  - ATtiny15
- **SPECIAL PURPOSE**
  - AT90PWM216 Especially for PWM
  - AT90USB1287 Especially for USB

megaAVR — the ATmega series
4–512 kB program memory
28–100-pin package
Extended instruction set (multiply instructions and instructions for handling larger program memories)
Extensive peripheral set

tinyAVR — the ATtiny series
0.5–16 kB program memory
6–32-pin package
Limited peripheral set

Příklad vlastností **ATmega32**

32K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities

1024 bytes EEPROM

2K byte SRAM

32 general purpose I/O lines

JTAG interface for Boundary scan

On-chip Debugging support and programming

3 flexible Timer/Counters with compare modes

Internal and External Interrupts

Serial programmable USART

Byte oriented Two-wire Serial Interface (I2C)

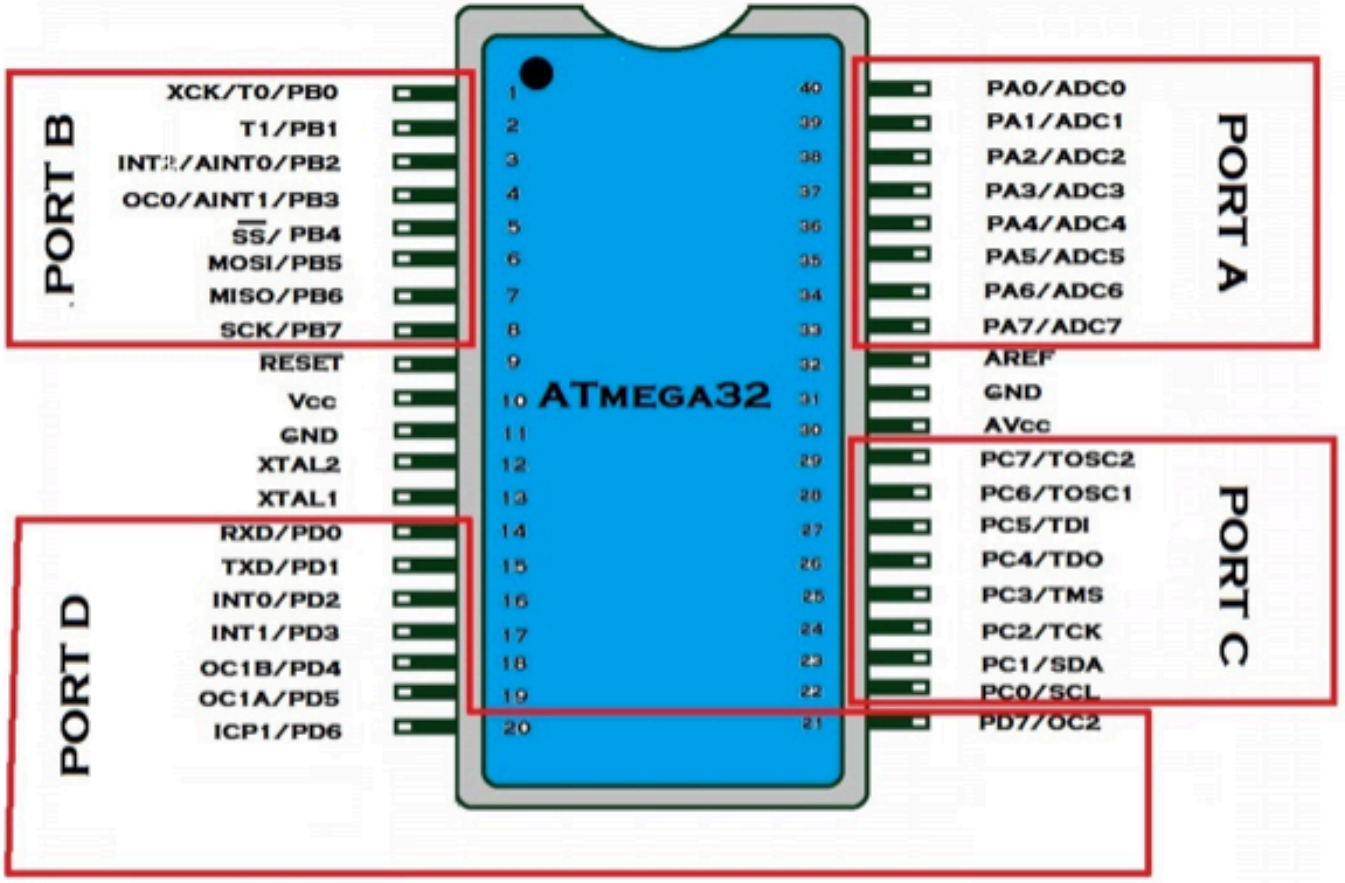
8-channel, 10-bit ADC

Programmable Watchdog Timer with Internal Oscillator

SPI serial port

6 software selectable power saving modes

Pinout Description of AVR Microcontroller



Port A (PA7..PA0): - Port A slouží jako analogové vstupy do převodníku A / D. Port A také slouží jako 8bitový obousměrný I / O port, pokud není použit převodník A / D. Portové kolíky. Portové piny mají metrické charakteristiky pohonu s vysokou propustností a schopností zdroje. Pokud jsou jako vstupy použity kolíky PA0 až PA7 a jsou externě vytáhnuty nízko, budou zdrojem proudu, pokud je vnitřní výsuv.

Port B (PB7..PB0): - Port B je 8-bitový obousměrný I / O port s vnitřními pull-up odpory (vybrány pro každý bit).

Port C (PC7..PC0): - Port C je 8-bitový obousměrný I / O port s vnitřními pull-up odpory (vybrány pro každý bit).

Port D (PD7..PD0): - Port D je 8bitový obousměrný I / O port s vnitřními pull-up odpory (vybrány pro každý bit).

RESET: - Resetovat vstup. Nízká úroveň na tomto pinu, která je delší než minimální délka impulsu, vygeneruje reset, a to i v případě, že hodiny neběží. Minimální délka impulsu je v definovaných impulsech, které nezaručují, že generují reset.

XTAL1: - Vstup do invertorového zesilovače oscilátoru a vstup do vnitřního provozního obvodu hodin.

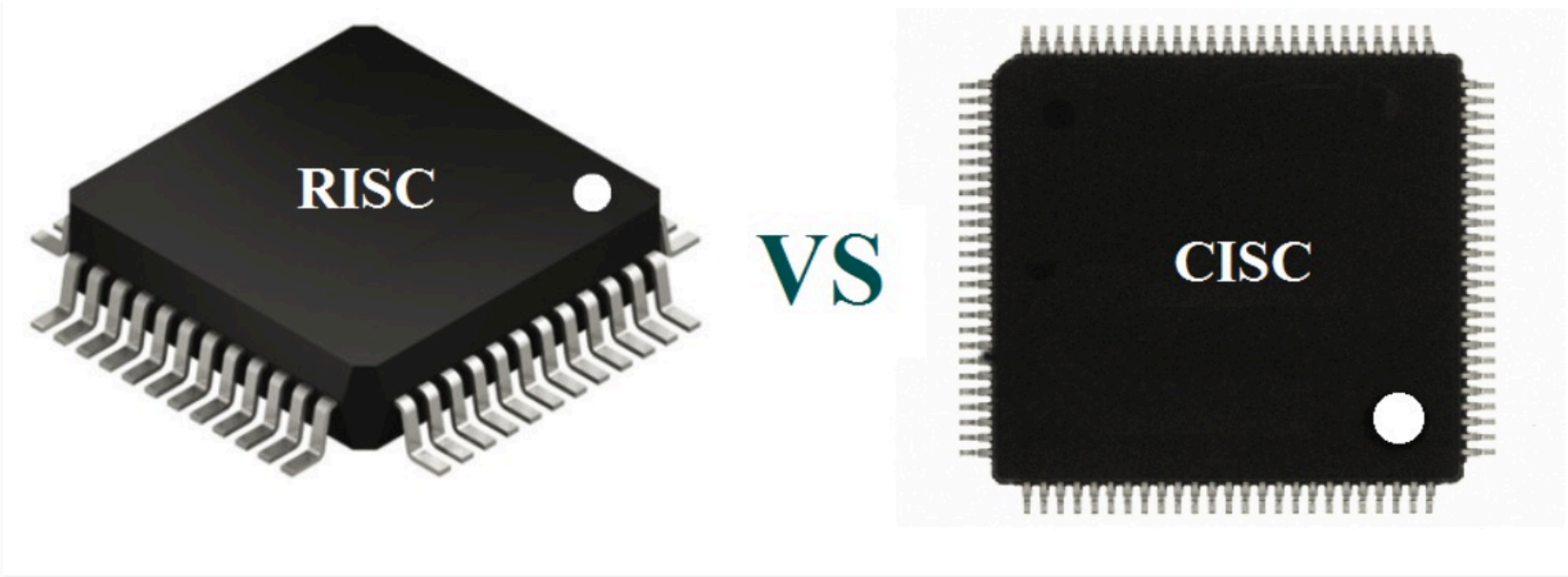
XTAL2: Výstup z invertorového zesilovače oscilátoru.

AVcc: - AVcc je pin napájecího napětí pro port A a převodník A / D.

AREF: - AREF je analogový referenční pin pro převodník A / D.

II. ČÁST

Rozdíl mezi architekturou RISC a CISC



CO JE CISC

Složitý počítač se sadou instrukcí (CISC - čteno sisk) je počítač, kde jednotlivé pokyny mohou provádět několik operací na nízké úrovni (například zatížení z paměti, aritmetické operace a paměťové úložiště) nebo jsou schopné multi-krokové operace nebo režimy adresování v rámci jednotlivých pokynů, jak jej název naznačuje "COMPLEX INSTRUCTION SET".

CO JE RISC

Počítač se sníženou sadou instrukcí (RISC – čteno „risk“) je počítač, který používá pouze jednoduché instrukce, které lze rozdělit do několika instrukcí, které provádějí nízkoúrovňový provoz v rámci jednoho cyklu hodin, jak naznačuje název "REDUCED INSTRUCTION SET“

PŘÍKLAD, JAK NA TO JDE CISC A RISC

Příklad **vynásobení dvou čísel A = A * B; <<< ====== Toto je prohlášení C**

Přístup CISC : Primárním cílem architektury CISC je dokončit úkol v co nejmenším počtu sestav. Toho je dosaženo budováním procesorového hardwaru, který je schopen porozumět a provádět sérii operací, to je místo, kde byla představena naše architektura CISC.

Pro tento konkrétní úkol by byl procesor CISC připraven se specifickou instrukcí (nazveme to "MULT"). Po provedení této instrukce

Načíst dvě hodnoty do samostatných registrů

Násobí operandy v prováděcí jednotce

A konečně třetí, ukládá produkt do příslušného registru.

Celý úkol vynásobení dvou čísel lze tedy doplnit **jednou instrukcí:**

MULT A, B <<< ====== Toto je zápis instrukce

MULT je to, co je známé jako "složitá instrukce". Operuje přímo na paměťových kartách počítače a nevyžaduje, aby programátor výslovně volal jakékoliv funkce nakládání nebo ukládání.

PŘÍKLAD, JAK NA TO JDE CISC A RISC

Příklad vynásobení dvou čísel A = A * B; <<< ====== Toto je prohlášení C

Přístup RISC : -Procesory RISC používají pouze jednoduché instrukce, které lze provést v rámci jednoho cyklu hodin. Příkaz "MULT" popsaný výše může být rozdělen do tří samostatných příkazů:

"LOAD", která přesune data z paměťové banky do registru
"PROD", který nalezne produkt dvou operandů umístěných v registrech
"STORE", která přesune data z registru do paměti.
Za účelem provedení přesné série kroků popsaných v přístupu CISC by programátor musel kódovat čtyři linky sestavení:

LOAD R1, A <<< ====== Toto je zápis instrukce

LOAD R2, B <<< ====== Toto je zápis instrukce

PROD A, B <<< ====== Toto je zápis instrukce

STORE R3, A <<< ====== Toto je zápis instrukce

Zpočátku se to může zdát mnohem méně účinným způsobem dokončení operace. Protože existuje více řádků kódu, je zapotřebí více paměti RAM pro uložení instrukcí na úrovni sestavy. Kompilátor také musí vykonat více práce, aby převedl jazykový výpis na vyšší úrovni do kódu tohoto formuláře.

CISC & RISC

Výhoda CISC:

- Kompilátor musí udělat velmi málo práce, aby překládal jazykové prohlášení na vysoké úrovni do sestavy
- Délka kódu je relativně krátká
- Pro ukládání pokynů je zapotřebí velmi málo paměti RAM
- Důraz je kladen na vytváření komplexních

Výhoda RISC:

- Každá instrukce vyžaduje pouze jeden cyklus hodin, který se provede, celý program bude spuštěn v přibližně stejném čase jako příkaz "MULT" s více cykly.
- Tyto "omezené instrukce" RISC vyžadují méně tranzistorů hardwarového prostoru než složité instrukce, což ponechává více prostoru pro registry pro všeobecné účely. Protože všechny instrukce se provádějí v jednotném množství času (tj. Jeden hodin)
- Pipeling je možný.

	ARM	8051	AVR	PIC	MSP430
Bus Width	32-bit mostly also available in 64-bit	8-bit for standard core	8/32-bit	8/16/32-bit	16-bit
Communication Protocols	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI (serial audio interface), IrDA	UART, USART, SPI, I2C	UART, USART, SPI, I2C, (special purpose AVR support CAN, USB, Ethernet)	PCI, UART, USART, LIN,CAN, Ethernet, SPI, I2S	UART, USART, LIN, I2C, SPI, I2S, IrDA
Speed	1 clock / instruction cycle	12 clock / instruction cycle	1 clock / instruction cycle	4 clock / instruction cycle	6 clock / instruction cycle
Memory	Flash, SDRAM, EEPROM	ROM, SRAM, FLASH	Flash, SRAM, EEPROM	SRAM, FLASH	SRAM, FLASH
ISA	RISC	CISC	RISC	Some feature of RISC	Some feature of RISC
Memory Architecure	Modified Harvard architecture	Von Neumann architecture	Modified Harvard	Harvard architecture	Von Neumann architecture
Power Consumption	Low	Average	Low	Low	Ultra Low
Families	ARMv4,5,6,7 and Cortex series	8051 variants	Tiny, Atmega, Xmega, special purpose AVR,	PIC16, PIC17, PIC18, PIC24, PIC32	MSP430X,MSP430FR57xx, MSP430x1xx to x6xx series
Community	Vast	Vast	Very Good	Very Good	Average
Manufacturer	Apple, Nvidia, Qualcomm, Samsung Electronics, and TI, etc.	NXP, Atmel, Silicon Labs, Dallas, Cyprus, infineon, etc	Atmel	Microchip	TI
Cost (as compared to feature provided)	Low	Very Low	Average	Average	Average
Other Feature	High speed operation	Known for its Standard	Cheap, effective	Cheap	Known for Ultra low power operation
Popular Microcontrollers	LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc	AT89C51, P89v51,etc	Atmega8,16,32, Arduino Community	PIC18fXX8, PIC16f88X, PIC32MXXX	MSP430G2553, MSP430 launchpad.

JAKÝ PROCESOR POUŽÍT ?

ARM - pokud vaše aplikace potřebuje řízení v reálném čase, rychlé zpracování, high-end komunikační protokol a mnoho dalších funkcí, jako je ADC, PWM spolu s velkou pamětí, pak ARM "Mighty Monster" se nejlépe hodí pro vaši aplikaci.

8051 - pokud vaše aplikace potřebuje méně funkce s levným rozpočtem na projekt a věrný spolehlivý produkt zvolit 8051, podle mých osobních zkušeností, 8051 je univerzální řešení, kde je levný rozpočet projektu a spolehlivý produkt s méně funkčností. Používal jsem 8051 jádro produktu v mnoha číslech aplikací, počínaje mým posledním ročníkem vysokoškolský projekt na high end stroj jako podpůrný řadič.

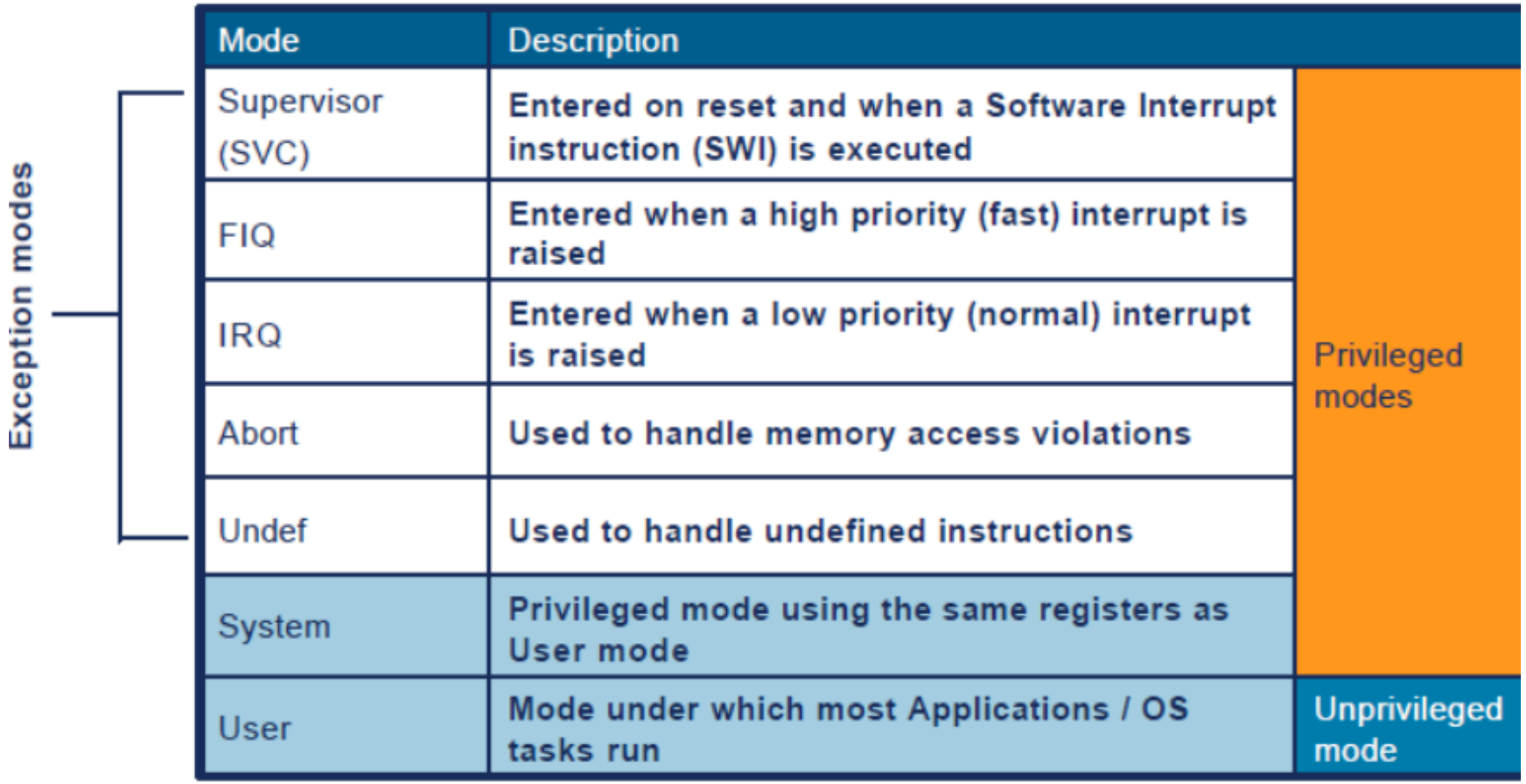
AVR - Viděl jsem tento regulátor ve většině odvětví robotiky a automobilového průmyslu kvůli jeho široké podpoře a jednoduchosti.

JAKÝ PROCESOR POUŽÍT ?

PIC - Používá se v průmyslových aplikacích AC, TV, lednice a mnoho dalších levných projektů. Vždycky jsem byl nenáviděný správci PIC kvůli jeho placené vývojové sadě systému a menší komunitní podpoře.

MSP430 - Co když říkám, že celý váš projekt bude fungovat na jediné baterii AA po dobu 5 let nebo vaše mikrokontrolér může pracovat i při napájecím napětí 0,9V. Možná nevěříte, ale je to možné s MSP430, nejlepším produktem společnosti TI

Processor Modes – Módy procesoru



ARM Instruction Set

All instructions are 32 bits long / many execute in a single cycle
Instructions are conditionally executed
A load / store architecture
Example **data processing instructions**

SUB r0,r1,#5	r0 = r1 − 5
ADD r2,r3,r3,LSL #2	r2 = r3 + (r3 * 4)
ADDEQ r5,r5,r6	IF EQ condition true r5 = r5 + r6

Example **branching instruction**

B <Label>	Branch forwards or backwards relative to current PC (+/- 32MB range)

Example **memory access instructions**

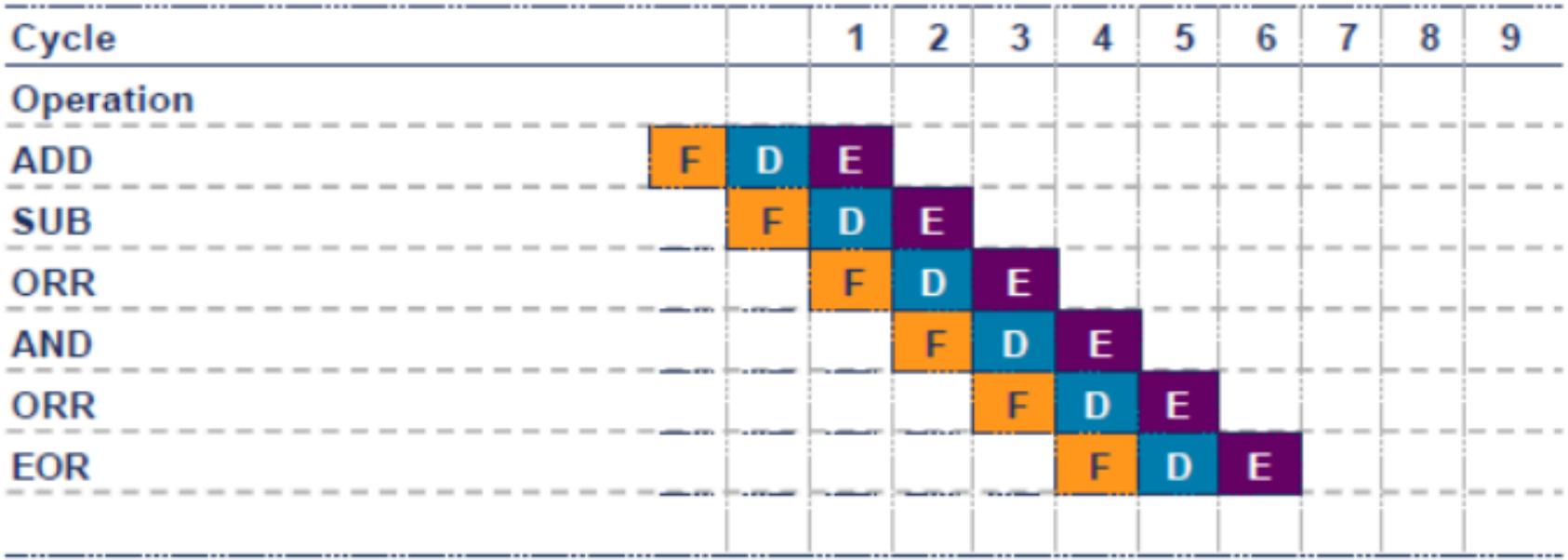
LDR r0,[r1]	Load word at address r1 into r0
STRNEB r2,[r3,r4]	IF NE condition true, store bottom byte of r2 to address r3+r4
STMFD sp!,{r4-r8,lr}	Store registers r4 to r8 and lr on stack. Then update stack pointer

Pipeline – „Pípování instrukcí“

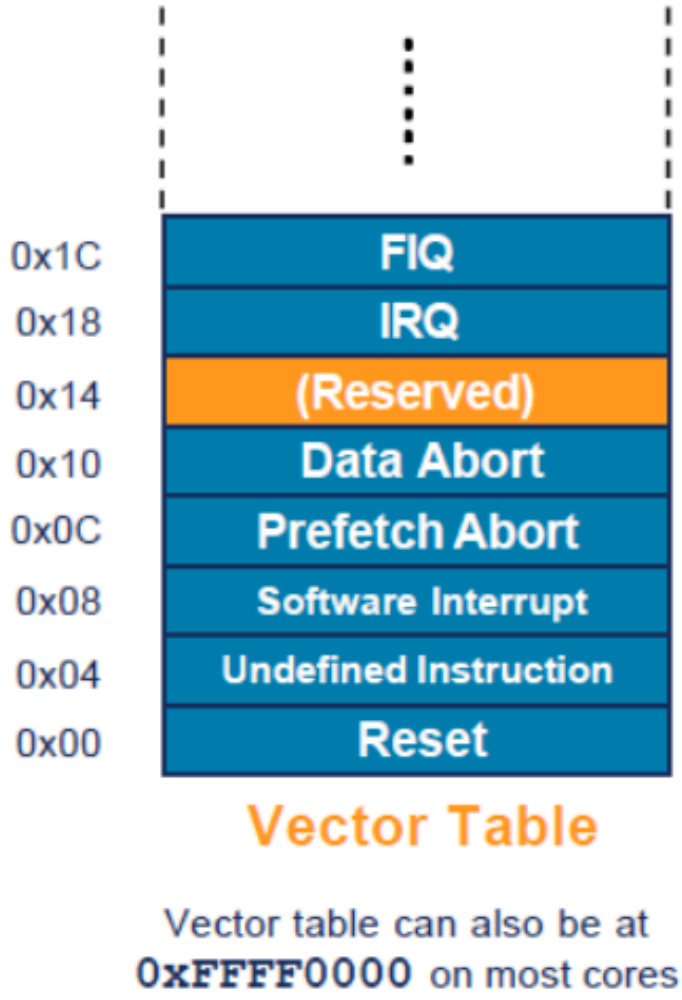
Fetch :- Instruction fetched from memory

Decode :- Decoding of registers used in instruction

Execute :- Register(s) read from Register Bank, Shift and ALU operation, Write register(s) back to Register Bank.



Exception Handling



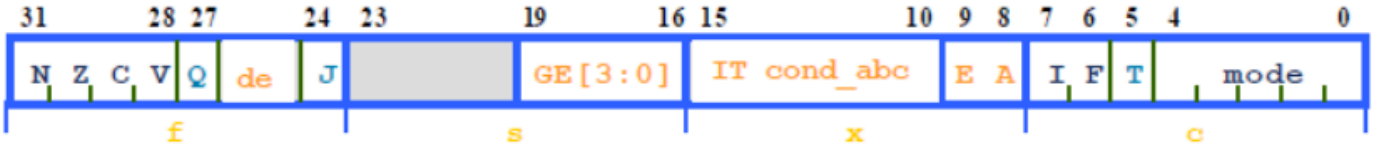
When an exception occurs, the core:

- Copies CPSR into SPSR_<mode>
- Sets appropriate CPSR bits
- Change to ARM state
- Change to exception mode
- Disable interrupts (if appropriate)
- Stores the return address in LR_<mode>
- Sets PC to vector address

To return, exception handler needs to:

- Restore CPSR from SPSR_<mode>
- Restore PC from LR_<mode>

CURRENT/STATE PROGRAM STATUS REGISTER (CPSR / SPSR)



Condition code flags (Bit 28 to 31)

- V = ALU operation o**V**erflowed
- C = ALU operation **C**arried out
- Z = **Z**ero result from ALU
- N = **N**egative result from ALU

Sticky Overflow flag – Q flag (Bit 27)

- Architecture 5TE and later only
- Indicates if saturation has occurred

J bit (Bit 24)

- Architecture 5TEJ and later only
- J = 1: Processor in Jazelle state

Interrupt Disable bits (Bit 6 Bit 7)

- I = 1: Disables IRQ**F = 1: Disables FIQ**

T Bit (Bit 5)

- T = 0: Processor in ARM state
- T = 1: Processor in Thumb state
- Introduced in Architecture 4T

Mode bits (Bit 0 to 4)

- Specify the processor mode

New bits in V6 (Bit 10 to 19)

- GE[3:0] used by some SIMD instructions
- E bit controls load/store endianness
- A bit disables imprecise data aborts
- IT [abcde] IF THEN conditional
- execution of Thumb-2 instruction groups

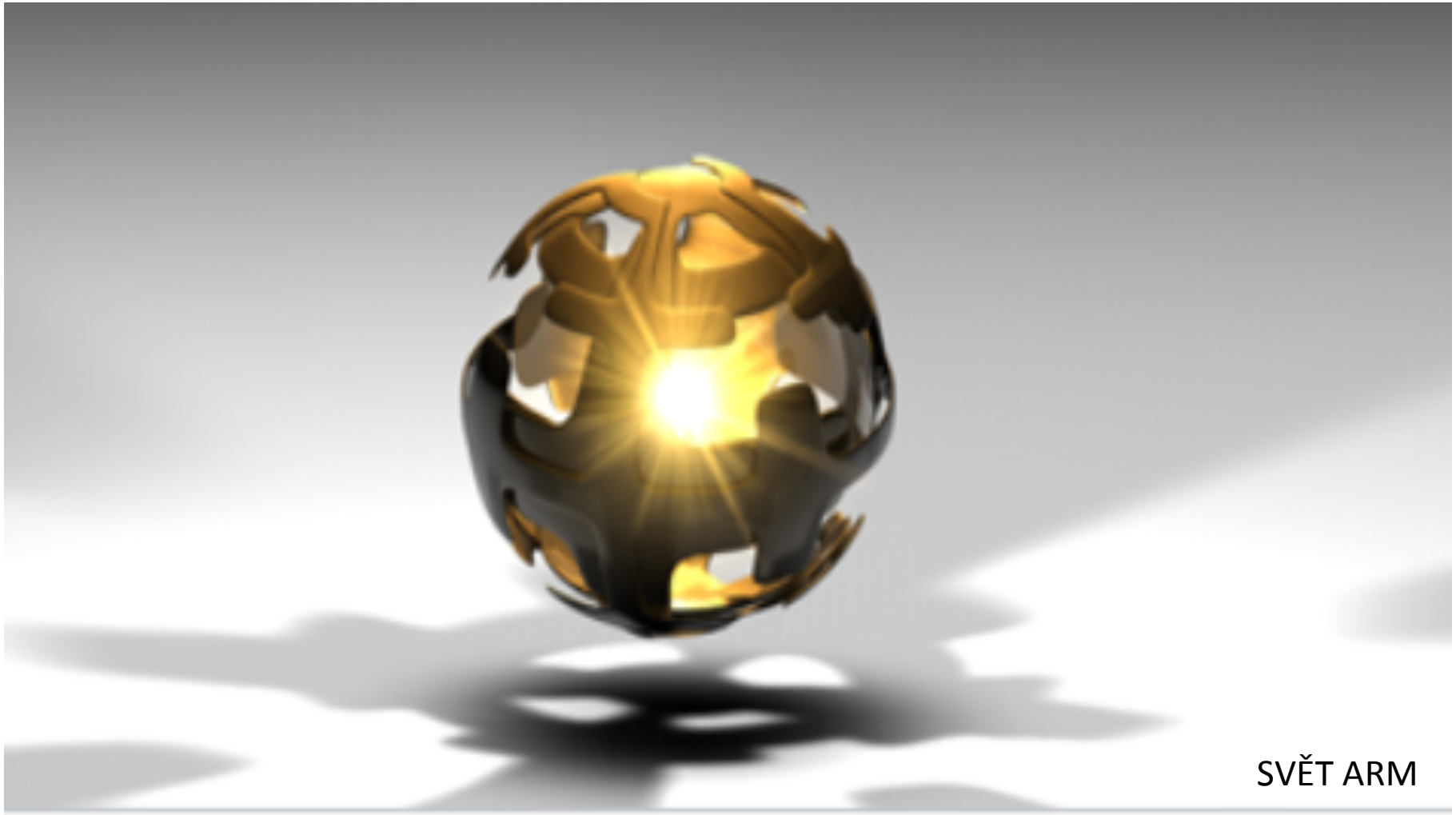
EPIC

EPIC (Explicitly Parallel Instruction Computing) : -EPIC je vynalezen společností Intel a je svým způsobem kombinací obou CISC a RISC. To bude teoreticky umožňovat zpracování aplikací založených na systému Windows i UNIX stejným procesorem.

Intel pracuje na něm pod kódovým jménem Merced. Společnost Microsoft již vyvíjí svůj standard Win64. Stejně jako název říká, Merced bude 64bitový čip.

Pokud je architektura Intel EPIC úspěšná, může to být největší podproces pro RISC. Všichni výrobci velkého procesoru, ale Sun a Motorola nyní prodávají produkty založené na platformě x86 a některé jen čekají, až Merced vyjde (HP, SGI). Vzhledem k trhu x86 není pravděpodobné, že CISC brzy zemře, ale RISC může.

III. POJMY



SVĚT ARM

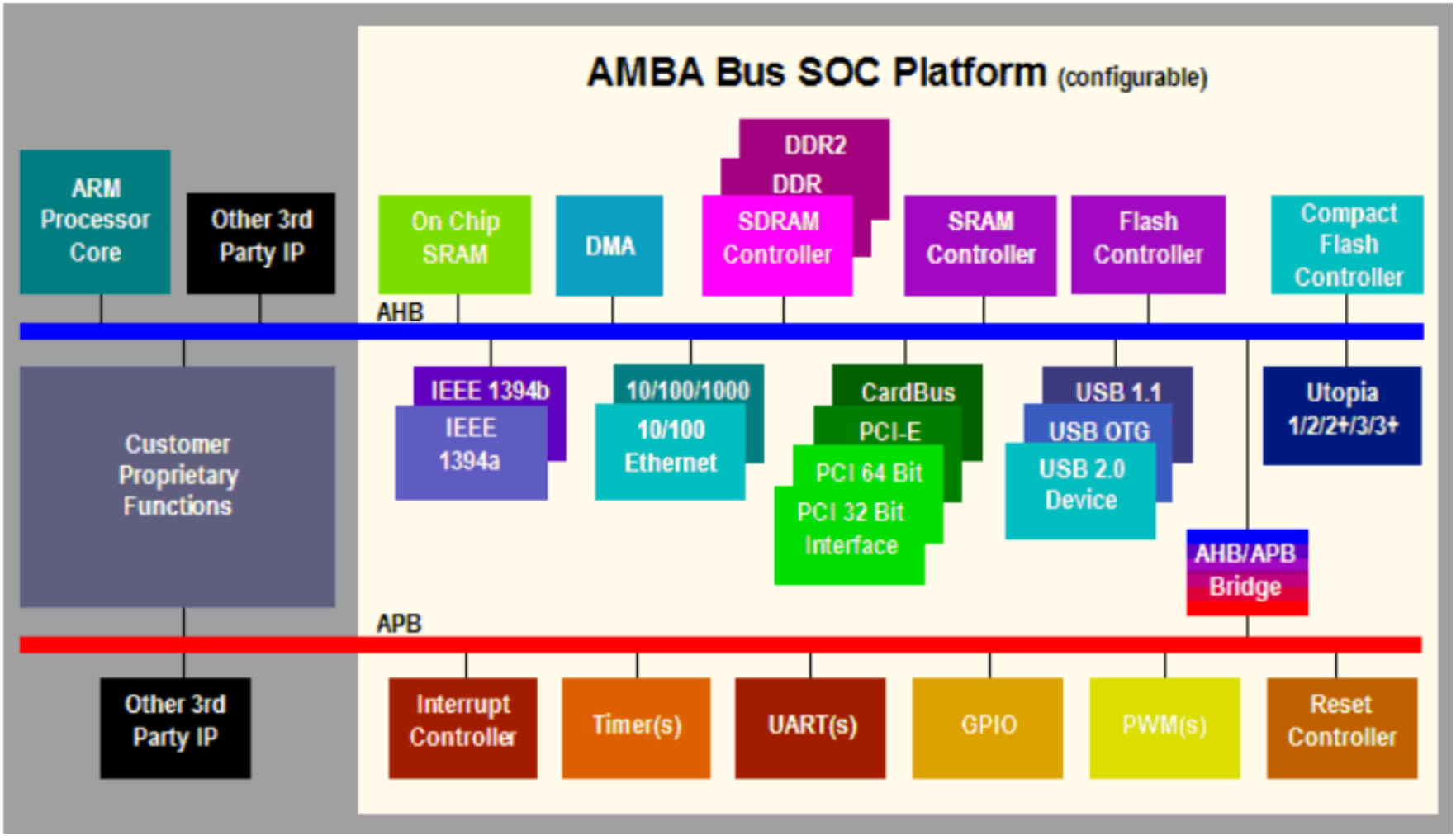
EMBEDDEDICE

Za účelem poskytnutí silného prostředí pro ladění pro aplikace založené na ARM byla logika **EmbeddedICE vyvinutá a integrována do základní architektury ARM**. Jedná se o soubor registrů, který umožňuje nastavit hraniční body hardwaru nebo body sledování kódu nebo dat. Logika EmbeddedICE monitoruje signály ARM jádra v každém cyklu, aby zkontrolovala, zda došlo k narušení bodu zlomu nebo hodinky.

Komunikace s logikou EmbeddedICE z vnějšího světa je poskytována prostřednictvím testovacího přístupového portu nebo TAP, řadiče a standardního připojení **IEEE 1149.1 JTAG.**

Výhodou řešení debugu na čipu je schopnost rychle ladit software, zejména pokud je software umístěn v ROM nebo Flash.

AMBA BUS
THE "ADVANCED MICRO-CONTROLLER BUS ARCHITECTURE"



KONSTRUKCE SPECIFIKACE SBĚRNICE AMBA

Konstrukce specifikace sběrnice AMBA je zaměřena na nízkou spotřebu energie a vysoký výkon. Typický systém **SoC** založený na technologii AMBA se skládá z pokročilé vysoce výkonné systémové sběrnice (**AHB)** a pokročilé nízké napájecí periferní sběrnice (**APB**). Jak je vidět na obrázku výše

Na výkonové kritické straně sběrnice je jádro ARM, řadič paměti, řídicí jednotka testovacího rozhraní (TIC), řadič LCD, IP on-chip, vlastní logika a speciální funkce.

Na straně s nízkým výkonem je rozhraní Smart Card, audio kodek, UART, PWM, časovače, GPIO atd.

To je vynikající příklad toho, jak autobusy AHB a APB pracují společně a poskytují kompletní systémové řešení.

Tyto protokoly sběrnice jsou nezávislé na procesoru ARM a jsou generalizovány pro aplikaci SoC. Metodika testu AMBA poskytuje mechanismus, který umožňuje externímu testeru přístup na sběrnici AMBA na čipu. To umožňuje, aby tester převzal kontrolu nad sběrnicí a zkontroloval každou součást samostatně.

MMU - THE MEMORY MANAGEMENT UNIT

Jednotka správy paměti pracuje se systémem vyrovnávací paměti pro ovládání přístupu do a z externí paměti. MMU také řídí překlad virtuálních adres na fyzické adresy a kontroly oprávnění přístupu pro instrukční a datové porty procesorů. MMU také zmírňuje problém fragmentace paměti.

MPU - MEMORY PROTECTION UNIT

Oddělená funkce **ochrany paměťových jednotek** byla zavedena do ARM pomocí jader Cortex-M3, které poskytují způsob, jak řídit přístupová práva paměti k aplikacím. To zabraňuje tomu, aby chyba v procesu ovlivňovala jiné procesy nebo samotný operační systém, a místo toho má za následek chybu segmentace nebo porušení výjimky paměti, což zpravidla způsobuje ukončení procesu. V jádře ARM jsou k dispozici samostatné registry, pomocí kterých můžete nakonfigurovat určitou část paměti a její přístupová práva.