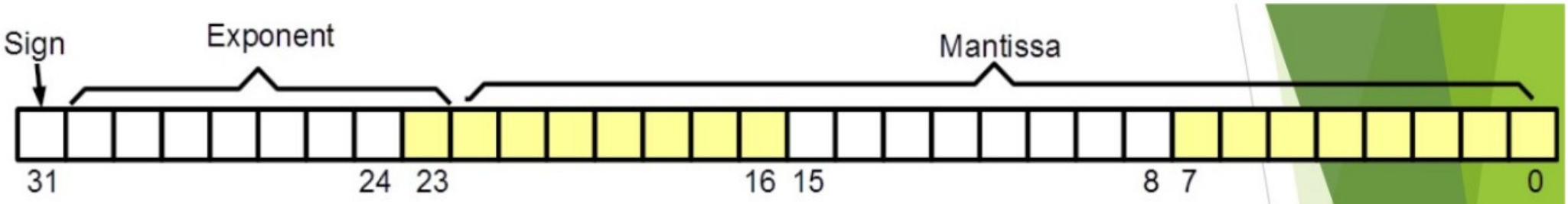
# Въведение в Cortex-A микропроцесорите

**Автор**: доц. д-р инж. Любомир Богданов

# Съдържание

# Преговор

От бакалавърския курс “Микропроцесорна схемотехника:




$$float = (-1)^{sign} \times 2^{exponent-127} \times \left(1 + \sum_{n=1}^{23} bit_{23-n} \times 2^{-n}\right)$$

# Преговор

**NaN** (Not A Number) – всички битове на експонентата са 1, а мантисата съдържа ненулева стойност. Полученото число е някоя от най-големите стойности, които дадения тип променлива може да представи. Битът за знак е без значение.

Използва се, за да представи непозволена операция:
0/0
∞/∞
√-x
0*∞
-∞+∞
∞-∞

# Преговор

**Signalling NAN** – NAN, който генерира прекъсване. За 32-битови числа от IEEE754-1985 стандартът, това са числата [a]:
*0x7F800001 – 0x7FBFFFFF или
*0xFF800001 – 0xFFBFFFFF

**Quiet NaN** – NAN, който не генерира прекъсване. За 32-битови числа от IEEE754-1985 стандартът, това са числата [a]:
*0x7FC00000 – 0x7FFFFFFF или
*0xFFC00000 – 0xFFFFFFFF

# Преговор

**Infinity** – числа, които се използват да покажат, че има грешка в изчислението – получената стойност е по-голяма (+∞) от най-голямата или по-малка (-∞) от най-малката стойност, която даденият тип променлива може да представи.

*Пример [b]:*
1/0 = +∞
log (0) = -∞

Експонента → всички битове = 1
Мантиса → всички битове = 0
Знак → 1 за -∞, 0 за +∞

# Преговор

За 32-битови числа от IEEE754-1985 стандартът, това са числата [a]:
$+\infty = 0x7F800000$
$-\infty = 0xFF800000$

# Въведение в микропроцесорите ARM Cortex-A

1. ARM Cortex-A микропроцесорите имат съкратен набор от инструкции (RISC)

2. Load-store (register-register) микроархитектура – инструкциите за изчисления могат да работят само върху регистровия файл на uPU, а не директно върху регистри от паметта/периферията

3. Прости режими на адресация, при които адресите се взимат от регистровия файл или са част от самата инструкция

# Въведение в микропроцесорите

| | Processor | | | | | |
|---|---|---|---|---|---|---|
| | **Cortex-A5** | **Cortex-A7** | **Cortex-A8** | **Cortex-A9** | **Cortex-A12** | **Cortex-A15** |
| Release date | Dec 2009 | Oct 2011 | July 2006 | March 2008 | June 2013 | April 2011 |
| Typical clock speed | ~1GHz | ~1GHz on 28nm | ~1GHz on 65nm | ~2GHz on 40nm | ~2GHz on 28nm | ~2.5GHz on 28nm |
| Execution order | In-order | In-order | In-order | Out of order | Out of order | Out of order |
| Cores | 1 to 4 | 1 to 4 | 1 | 1 to 4 | 1 to 4 | 1 to 4 |
| Peak integer throughput | 1.6DMIPS/ MHz | 1.9DMIPS/ MHz | 2DMIPS/MHz | 2.5DMIPS/MHz | 3.0DMIPS/ MHz | 3.5DMIPS/ MHz |
| VFP architecture | VFPv4 | VFPv4 | VFPv3 | VFPv3 | VFPv4 | VFPv4 |
| NEON architecture | NEON | NEONv2 | NEON | NEON | NEONv2 | NEONv2 |
| Half precision extension | Yes | Yes | No | Yes | Yes | Yes |
| Hardware Divide | No | Yes | No | No | Yes | Yes |
| Fused Multiply Accumulate | Yes | Yes | No | No | Yes | Yes |
| Pipeline stages | 8 | 8 | 13 | 9 to 12 | 11 | 15+ |
| Instructions decoded per cycle | 1 | Partial dual issue | 2 (Superscalar) | 2 (Superscalar) | 2 (Superscalar) | 3 (Superscalar) |

# Въведение в микропроцесорите ARM Cortex-A

Към 2024 процесорите от A-профила са (20 броя):

-A5, -A7, -A9, -A15, -A17, -A32, -A34, -A35,

-A53, -A55, -A57, -A65, -A72, -A73, -A75, -A76,

-A77, -A78, -A520, -A720

# Въведение в микропроцесорите ARM Cortex-A

1. ARM Cortex-A има три версии на набора от инструкции (към март 2024 г.):
*Armv7-A
*Armv8-A (= AArchv8-A)
*Armv9-A (= AArchv9-A)

2. Поддържат се 32- и 64-битови инструкции, съответно наборите се наричат:
*AArch32 (A32 + T32 = ARM32 + Thumb-II)
*AArch64 (A64 = ARM64, адресите се съхраняват в 64-битови регистри, а инструкциите могат да работят върху 64-битови величини, но дължината на инструкцията си остава 32-бита)

# Въведение в микропроцесорите ARM Cortex-A

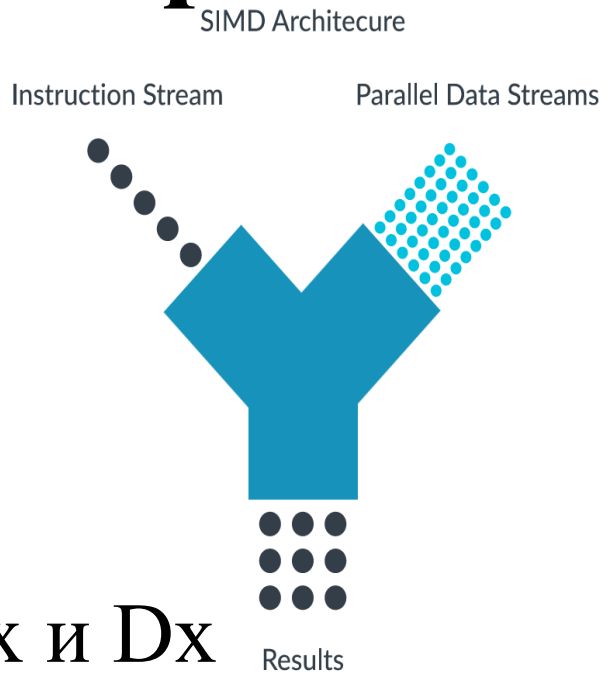
1. AArch32 включва инструкциите:

*SIMD инструкции, работещи с Rx регистри (32-битови цели числа)

*SIMD инструкции, работещи с Sx и Dx регистри на FPU (32- и 64-битови дробни числа)

*Скаларни инструкции, работещи с Sx и Dx регистри на FPU (32- и 64-битови дробни числа)

# Въведение в микропроцесорите ARM Cortex-A




1. AArch64 включва инструкциите:
*SIMD инструкции, работещи с Sx и Dx регистри на FPU
(32- и 64-битови дробни числа)

*Скаларни инструкции, работещи с Sx и Dx регистри на FPU (32- и 64-битови дробни числа)

*SVE (Scalable Vector Extension) инструкции, работещи с вектори с променлива дължина

***Няма** SIMD инструкции, работещи с Rx регистри (32-битови цели числа)

# Въведение в микропроцесорите ARM Cortex-A

1. ARM Cortex-A зависят изцяло от работа с MMU – Virtual Memory System Architecture (VMSA). Може да работи с 32- и 64-битови виртуални адреси.
   Набори: A64, A32, T32

2. ARM Cortex-R зависят изцяло от работа с MPU – Protected Memory System Architecture (PMSA).
   Набори: A64, A32, T32

3. ARM Cortex-M залагат на бързо обслужване на прекъсвания и отчасти на PMSA.
   Набори: T32

# Въведение в микропроцесорите ARM Cortex-A

Поддържани целочислени типове данни:

*byte – 8 бита

*halfword – 16 бита

*word – 32 бита

*double word – 64 бита

*quad word – 128 бита

# Въведение в микропроцесорите ARM Cortex-A

Поддържани типове данни за числа с плаваща запетая [1]:

*half-precision – 16 бита (IEEE 754-2008)

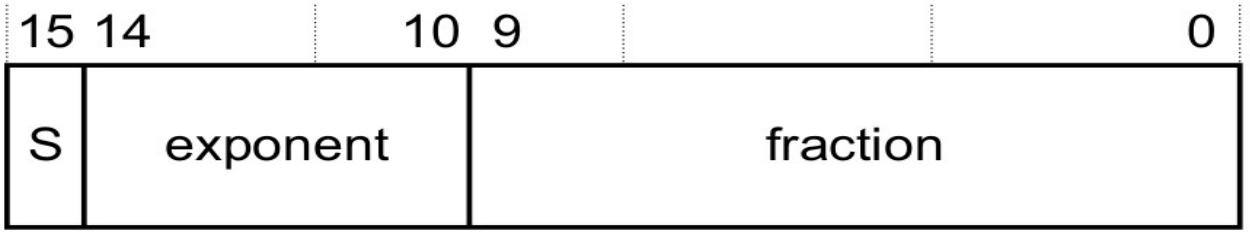
*single-precision – 32 бита (IEEE 754)

*double-precision – 64 бита (IEEE 754)

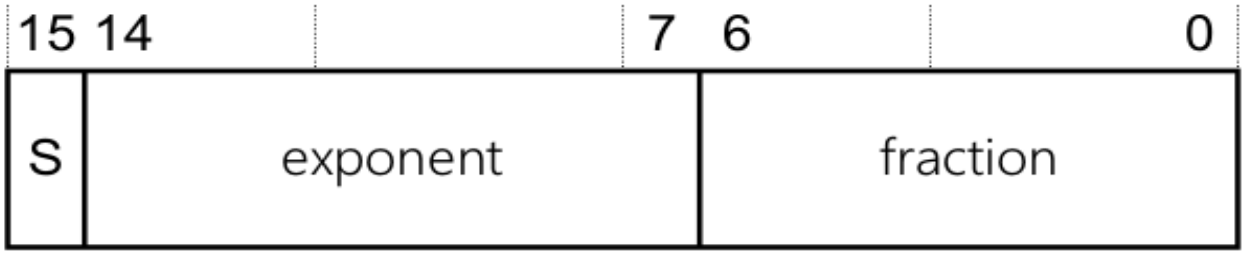
*BFloat16 – 16 бита (=single-precision) (brain floating point)

# Въведение в микропроцесорите ARM Cortex-A

*half-precision –




*BFloat16 –

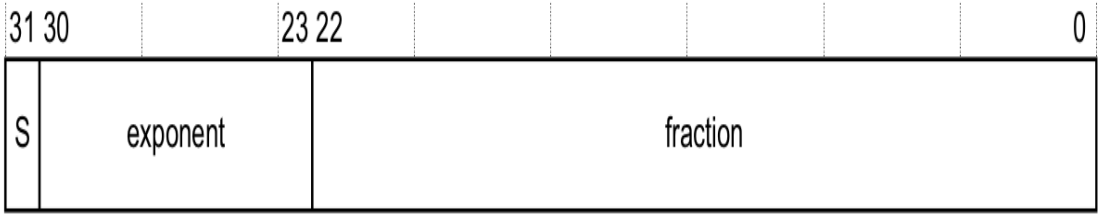


# Въведение в микропроцесорите ARM Cortex-A

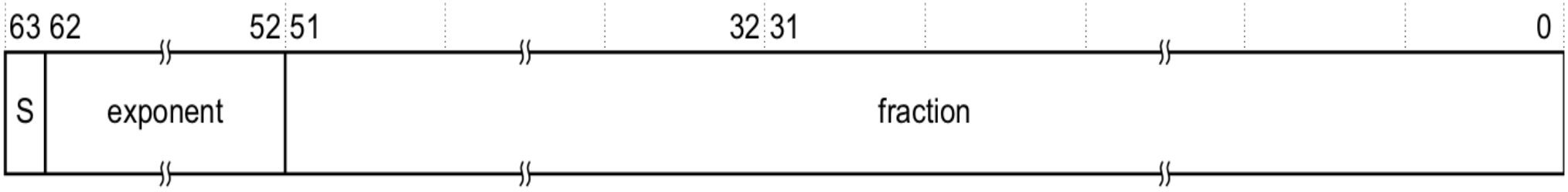
| | 0 < exp < 0x1f | | exp = 0 | | exp = 0x1f | |
|---|---|---|---|---|---|---|
| | **min** | **max** | **min** | **max** | **min** | **max** |
| **Half-precision** | $6{,}104.10^{-5}$ | 65504 | $5{,}960.10^{-8}$ | - | - | 131008 |
| | **0 < exp < 0x7f** | | **exp = 0** | | **exp = 0x7f** | |
| **BFloat16** | $1{,}175.10^{-38}$ | $3{,}390.10^{38}$ | $9{,}184.10^{-41}$ | - | - | - |

# Въведение в микропроцесорите ARM Cortex-A

*single-precision –




*double-precision –



# Въведение в микропроцесорите ARM Cortex-A

Поддържани типове данни за числа с **фиксирана** запетая:

* 32-битови стойности от Wx или Rx регистър

*64-битови стойности от Xx регистър

# Въведение в микропроцесорите ARM Cortex-A

AArch32 използва регистров файл с 32 64-битови регистъра (различни от Integer регистровия файл) за векторни или скаларни FPU изчисления.

Вектори (SIMD инструкции):
*16 128-битови quad word регистри, **Q0-Q15**
*32 64-битови double word регистри, **D0-D31**

Скаларни числа с плаваща запетая (FPU инструкции):
*32 32-битови single word регистри, **S0-S31**
* 32 64-битови double word регистри, D0-D31

# Въведение в микропроцесорите ARM Cortex-A

AArch64 използва регистров файл с 32 128-битови регистъра (различни от Integer регистровия файл) за векторни или скаларни FPU изчисления.

Вектори (SIMD инструкции):
Брой вектори n = 0 – 31
128-битови вектори: **Vn**{.2D, .4S, .8H, .16B}
64-битови вектори: **Vn**{.1D, .2S, .4H, .8B}.
B = 8 bits
H = 16 bits
S = 32 bits
D = 64 bits

# Въведение в микропроцесорите ARM Cortex-A

Това **подмножество** инструкции се нарича **NEON** – допълнителни инструкции (**extension** към AArch32, AArch64) с вектори с фиксирана дължина (32 128-битови регистъра с елементи 8, 16, 32, 64) [4].

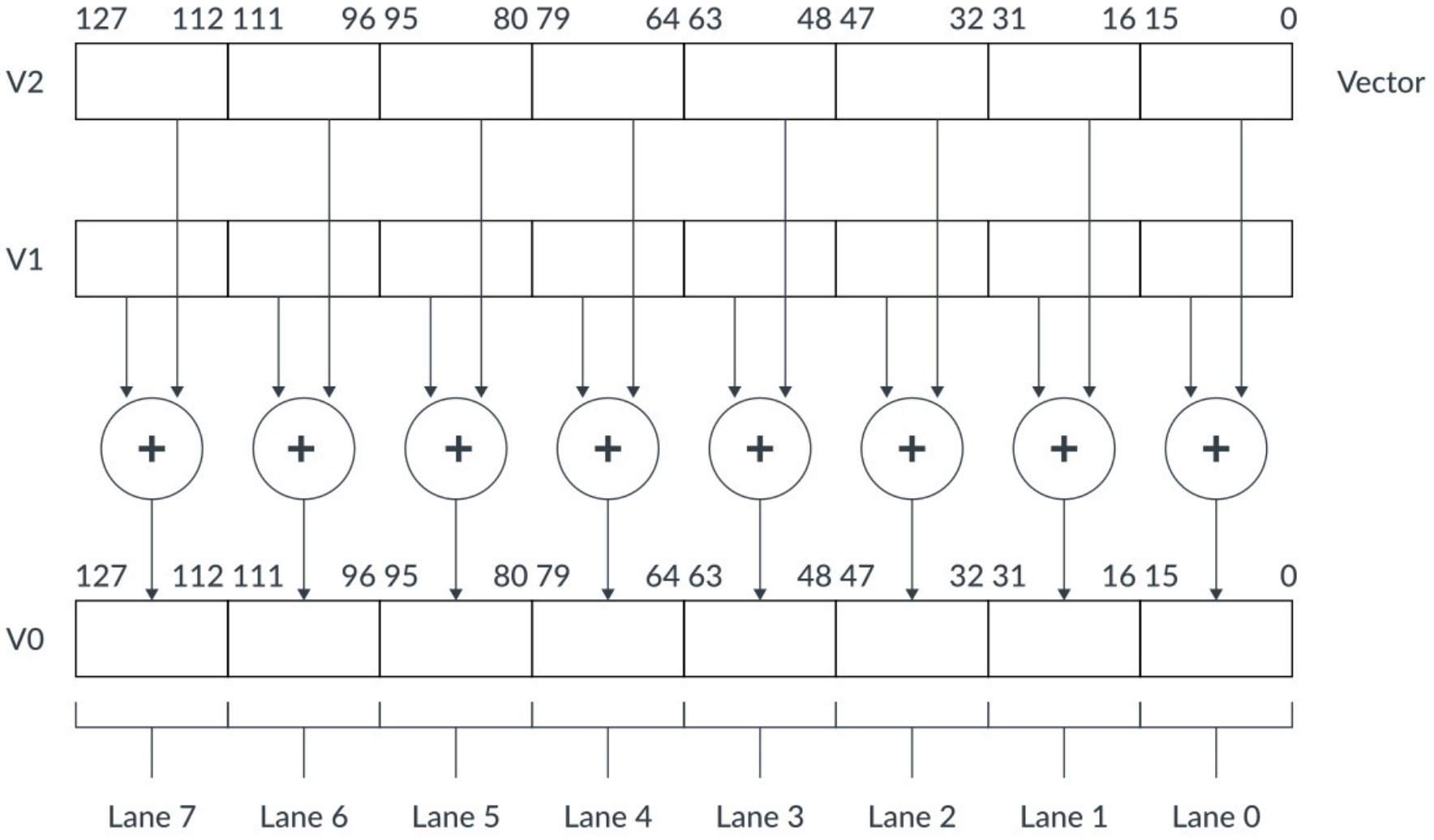
ADD **V0**.8H, **V1**.8H, **V2**.8H (събери 8 16-битови числа от V1 с 8 16-битови числа от V2 и запиши резултатът във V0)

MUL **V0**.4S, **V2**.4S, **V3**.S[2] (умножи 4 32-битови числа от V2 с 1 32-битово число от позиция 2 на V3 и запиши резултатът във V0)

# Въведение в микропроцесорите ARM Cortex-A

Figure 3-2: Order of elements in a vector

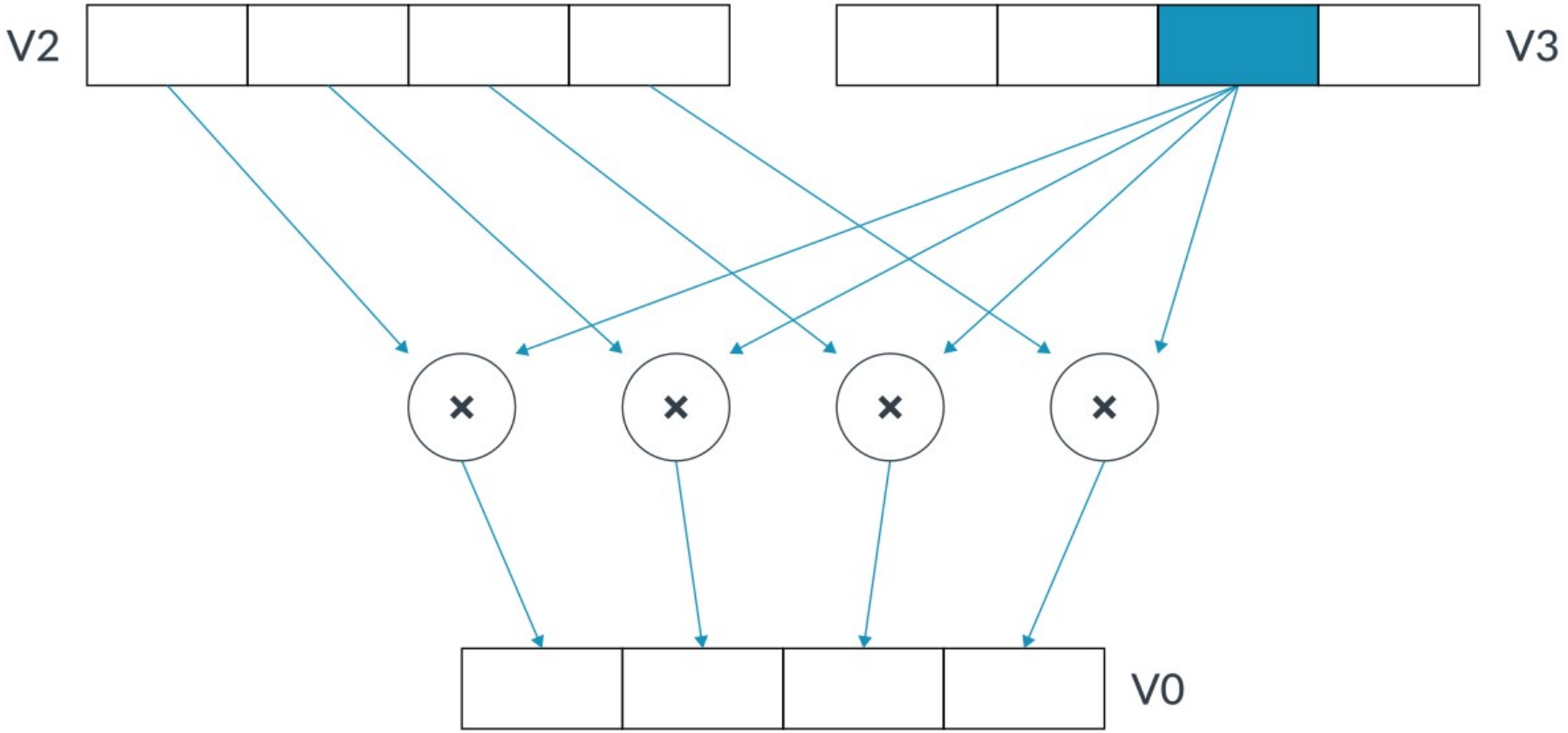
[4]



# Въведение в микропроцесорите ARM Cortex-A

[4]

Figure 3-3: Instructions using scalars



# Въведение в микропроцесорите ARM Cortex-A

AArch64 включва SVE (Scalable Vector Extension) – допълнителни инструкции с вектори с **променлива дължина**. Дължината варира в обхват:

128 – 2048 бита

и е кратна на 128 бита.

Регистрите, за такива вектори се отбелязват в инструкциите със **Z0 - Z31**.

# Въведение в микропроцесорите ARM Cortex-A

Инструкции за FP (Floating Point) или за векторна SIMD (Single Instruction Multiple Data) обработка на числа с плаваща запетая могат да генерират следните прекъсвания (наричани в документацията на ARM – изключения, exceptions):

*Input Denormal (IDE)
*Inexact (IXE)
*Underflow (UFE)
*Overflow (OFE)
*Divide by Zero (DZE)
*Invalid Operation (IOE)

Разрешават се от регистър FPCR и имената на битовете са показани в скобите по-горе.

# Въведение в микропроцесорите ARM Cortex-A

***Input Denormal (IDE) –** експонентата е нула, мантисата е ненулева стойност, получават се много малки числа, близки до нула. *(За разлика от Underflow, вж сл. слайд, тези числа все още могат да бъдат представени с floating point / double тип променлива, но работата с тях отнема повече тактове на FPU модула).*

***Inexact (IXE) –** резултатът от изчислението се различава от истинската стойност, поради ограничение в разредността на FPU модула (напр. 1/3 = 0.33333(3)).

# Въведение в микропроцесорите ARM Cortex-A

***Underflow (UFE) –** резултатът от изчислението е число близко до нула, което е по-малко от най-малката стойност, която може да бъде представена с избраната разредност на числото с плаваща запетая. (малко число / голямо число).

***Overflow (OFE)** - резултатът от изчислението е много голямо число, което е по-голямо от най-голямата стойност, която може да бъде представена с избраната разредност на числото с плаваща запетая. ( голямо число * голямо число).

# Въведение в микропроцесорите ARM Cortex-A

***Divide by Zero (DZE)** – генерира се от инструкция, която се опита да извърши делене на крайно ненулево число с нула.

# Въведение в микропроцесорите ARM Cortex-A

***Invalid Operation (IOE)** – генерира се от инструкция, която се опита да извърши една от следните операции:

→ поне един операнд е NaN (Not a Number)

→ изваждане на числа със стойност “безкрайност” (infinite)

→ делене на нула с безкрайност

→ делене на нула с нула

→ делене на безкрайност с безкрайност

→ коренквадратен на число с отрицателна стойност

# Програмен модел ARM Cortex-A

<u>За AArch64</u>

* 31 64-битови регистъра с общо предназначение. Отбелязват се с **X0 – X30.**

X30 – link register – за съхранение на адрес на връщане от подпрограма.

* Xx регистрите може да бъдат третирани като 31 32-битови регистъра. Отбелязват се с **W0 – W30**.

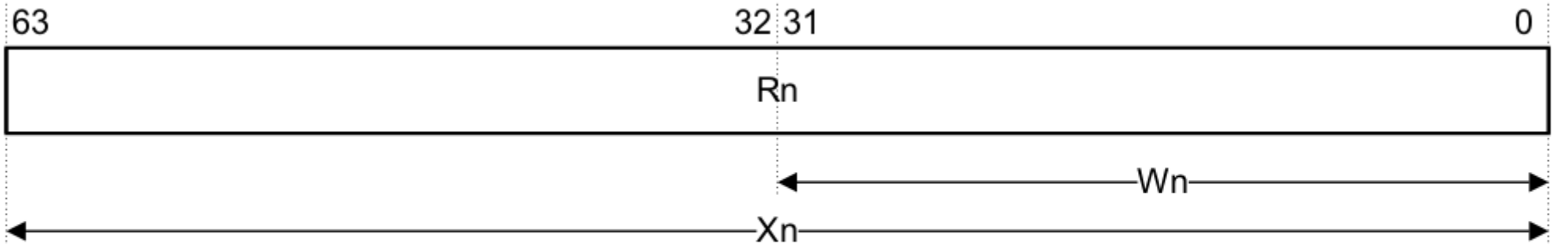



**Figure B1-1 General-purpose register naming**

# Програмен модел ARM Cortex-A

За AArch64

* Инструкциите имат адрес за регистър X31, който се води нулев регистър (ZR, zero register) и реално не съществува.

* Този адрес съдържа само една стойност – това е 0. Не може да бъде записван. Може да бъде четен.

* Използва се за занулляване (и други операции с нула) на други регистри, като ползата от него е, че не се изисква изтегляне (fetch) на константа от паметта.

# Програмен модел ARM Cortex-A

За AArch64

4. 64-битов програмен брояч PC – **не може** да бъде директно записван от инструкции. Може да бъде променян само:

* от branch инструкции
* при влизане в прекъсване
* при излизане от прекъсване

Опит за неподравнен достъп до инструкция ще генерира PC alignment fault прекъсване

5. 64-битов стеков указател SP (младшите 32-бита са достъпни с името WSP)

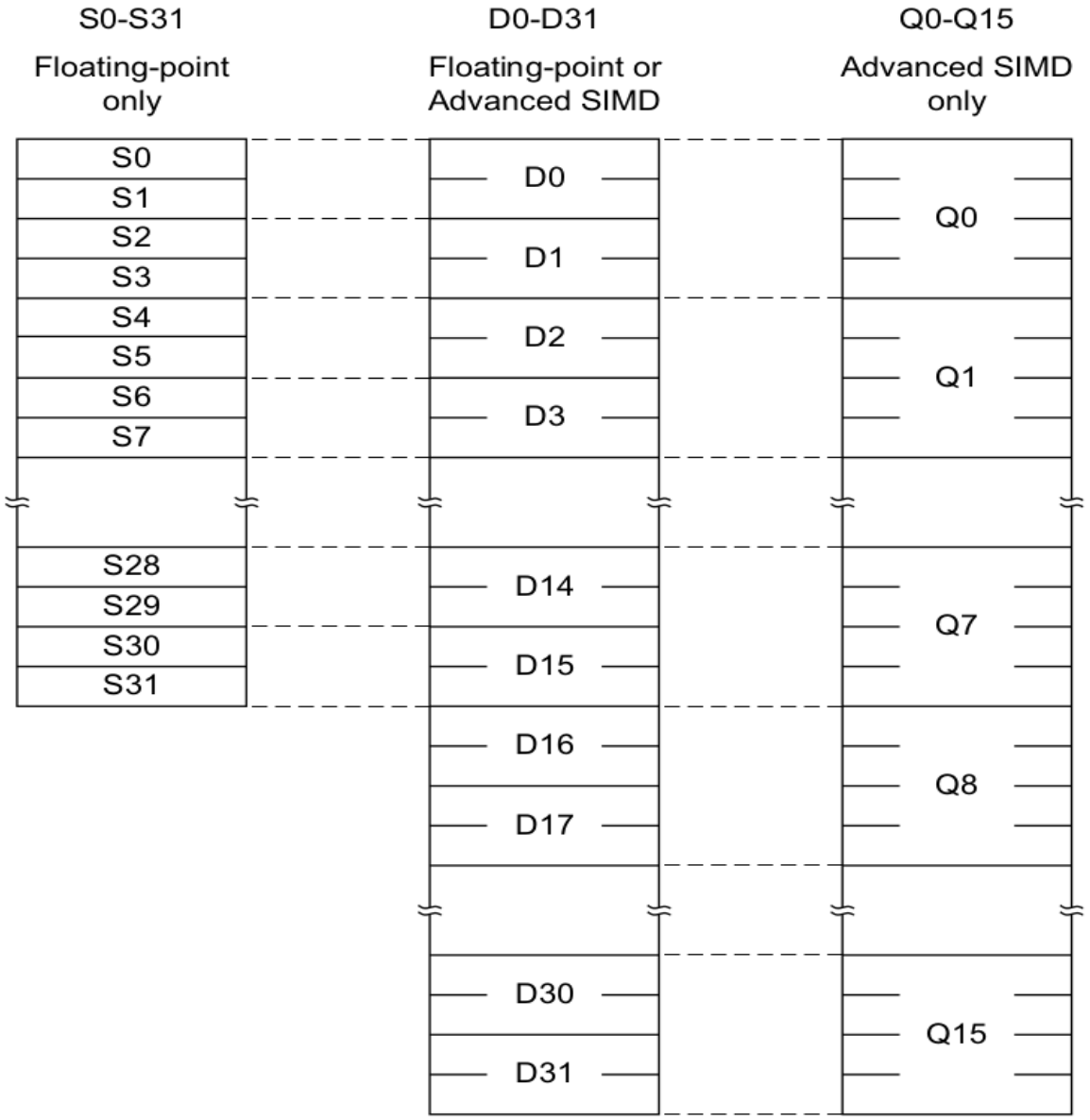
# Програмен модел ARM Cortex-A

6. 64-битов link регистър за съхранение на адрес на връщане при излизане от прекъсване (Exception Link Register)
7. 32 128-битови регистъра за векторни SIMD и скаларни FPU инструкции с общо наименование **V0-V31**. При SIMD могат да бъдат разделени на:

*Z0 – Z31 (128-2048-bit vector)
*V0-V31 (128-bit vector)
*Q0 – Q31 (128-bit)
*D0 – D31 (64-bit)
*S0 – S31 (32-bit)
*H0 – H31 (16-bit)
*B0 – B31 (8-bit)

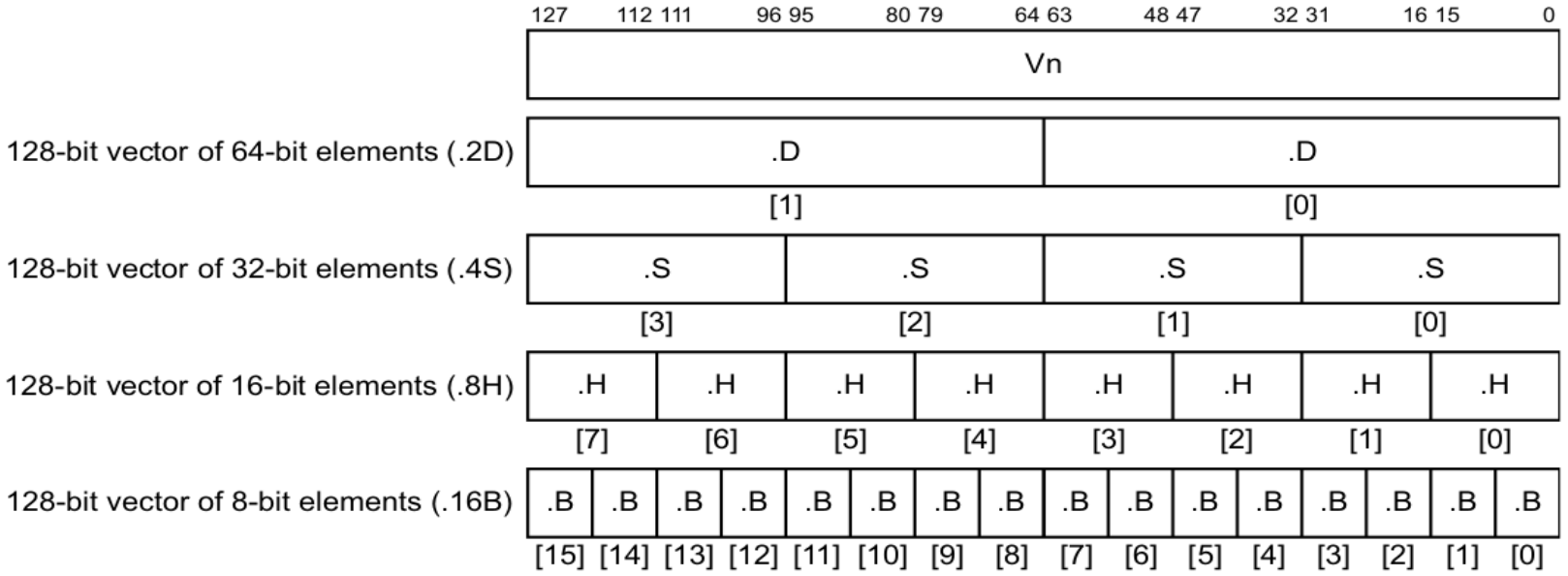
# Програмен модел ARM Cortex-A

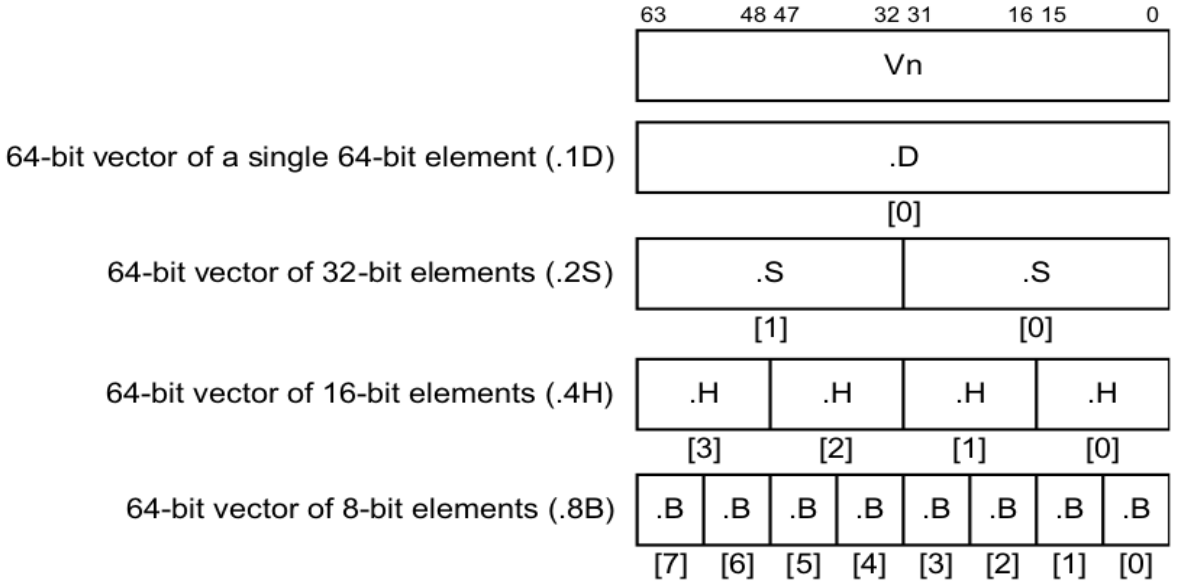
AArch32



# Въведение в микропроцесорите ARM Cortex-A

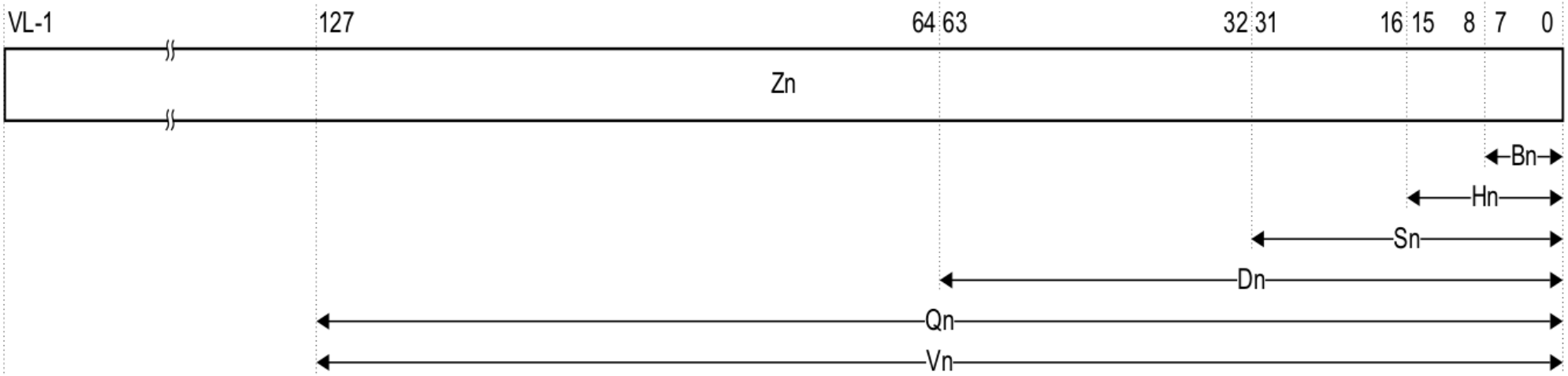
AArch64



# Въведение в микропроцесорите ARM Cortex-A

AArch64
(SVE инструкции)



# Програмен модел ARM Cortex-A

8. FPCR (Floating Point Control Register) – контролен регистър на FPU модула.

9. FPSR (Floating Point Status Register) – регистър със статус битове на FPU модула.

# Програмен модел ARM Cortex-A

За AArch32

1.13 32-битови регистъра с общо предназначение, отбелязвани с **R0 – R12**

2. 32-битов програмен брояч

3. 32-битов стеков указател

4. 32-битов link регистър за съхранение на адреса на връщане от подпрограма или от прекъсване.

5. 32-битов link регистър за връщане от Hyp mode*

6. 32 64-битови регистъра за векторни SIMD инструкции и скаларни FPU инструкции

*виж виртуализация

# Програмен модел ARM Cortex-A

ARM Cortex-A позволяват да се преминава от AArch64 към AArch32, което се нарича *interprocessing*.

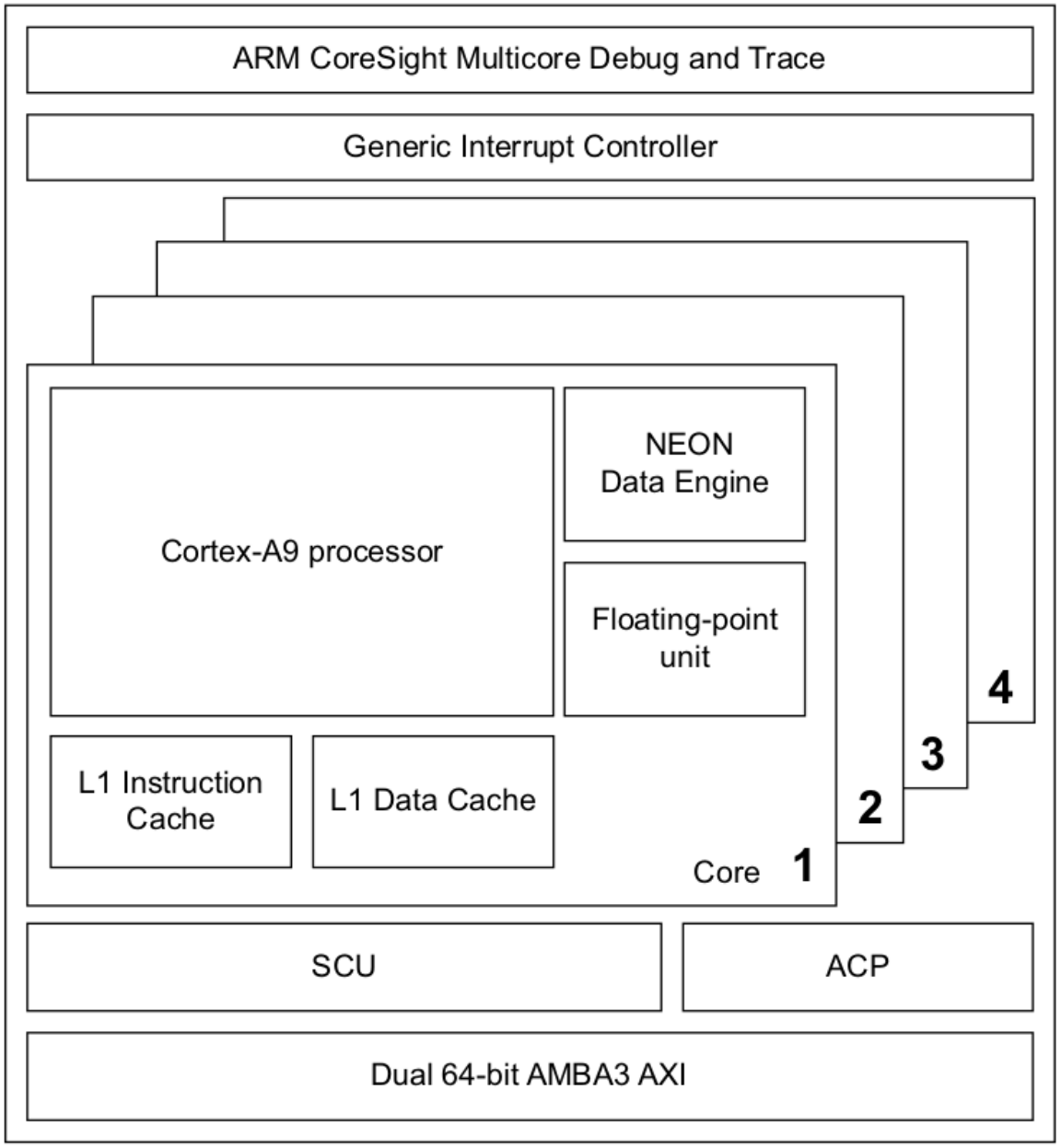
Интероперацията може да стане само при излизане от прекъсване.

# Програмен модел на ARM Cortex-A периферия

Ядрата на всички Cortex-A имат периферни модули, които се считат за част от микропроцесора:

* модул за системен контрол (SCB)
* контролер на прекъсванията (GIC)
* системен таймер (SysTick)
* модул за статистика (PMU - Performance Monitor Unit)
* модул за векторни изчисления (NEON)
* модул за дебъг и трасиране (CoreSight)
* модул за кеш-кохерентност между няколко ядра (SCU Snoop Control Unit)
* L1 + L2 кеш памет
* интерфейс за четене на кеша от външно устройство, напр. DMA (ACP – Accelerator Coherency Port)

# Програмен модел на ARM Cortex-A периферия



# Виртуализация

Достъпът до паметта при ARM Cortex-A се характеризира с:

* при достъпване на **неподравнен** (unaligned) адрес се генерира изключение/exception (=прекъсване от самия процесор, а не от периферия);
* приложните програми **нямат достъп** до определени адреси от паметта;
* има **транслиране** на виртуалните адреси (VA) към физически адреси (PA); физическите адреси се наричат още абсолютни адреси
* може да се преобразува между **big endian и little endian** представяне на данните;
* може да се **контролира последователността** на достъпите до паметта;

# Виртуализация

* може да има **кеш** памет;
*може да се синхронизира достъпа на няколко главни устройства до **споделена памет**;
*има **бариерни инструкции**, които преустановяват спекулативния достъп до паметта;

# Виртуализация

AArch64 и AArch32:
* 64-битово адресиране на виртуални адреси;

* може да има до **2 отделни виртуални адресни полета**, всяко от което си има независими контролни модули;

* виртуалните адреси се транслират във физически на блокове и страници навсякъде във физическото адресно поле [2].

# Виртуализация

* Ако дадено приложение изисква използването на:
→ две или повече операционни системи (например Linux и Windows)
→ два или повече софтуера с различни изисквания за изпълнение в реално време (например Linux и bare-metal firmware)
→ разделяне на софтуера на доверен/непроверен (за приложения изискващи сигурност)

Може да се използва един процесор, ако той поддържа **виртуализация**.

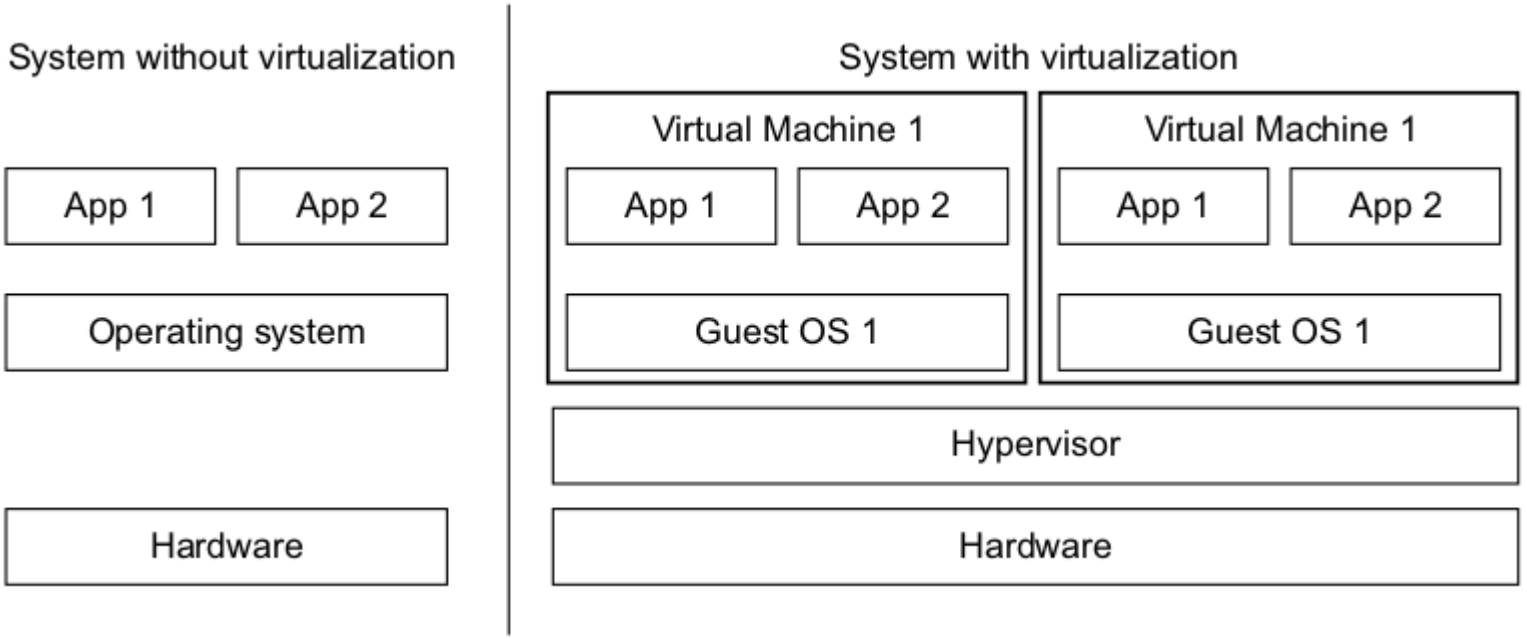
# Виртуализация

*За целта софтуер, наречен Hypervisor осигурява две (или повече) виртуални адресни полета, в които се изпълнява кода на различните софтуери. Hypervisor се среща още като Virtual Machine Monitor

*Достъпът до хардуера минава изцяло през Hypervisor-а.

*Hypervisor-а изисква хардуер с разширени възможности, за да направи виртуализацията (разделянето) възможна.

# Виртуализация



# Виртуализация

*В процесорите от профила ARM Cortex-A освен съществуващите в Cortex-M и -R режими “привилигирован” и “непривилигирован”, се добавя и още един **режим “хипервайзор”**.

*В MMU транслациите на адреси се добавя още едно ниво на транслиране:

VA → PA (MMU на CPU без виртуализация)
VA → IPA → PA (MMU на CPU с виртуализация)

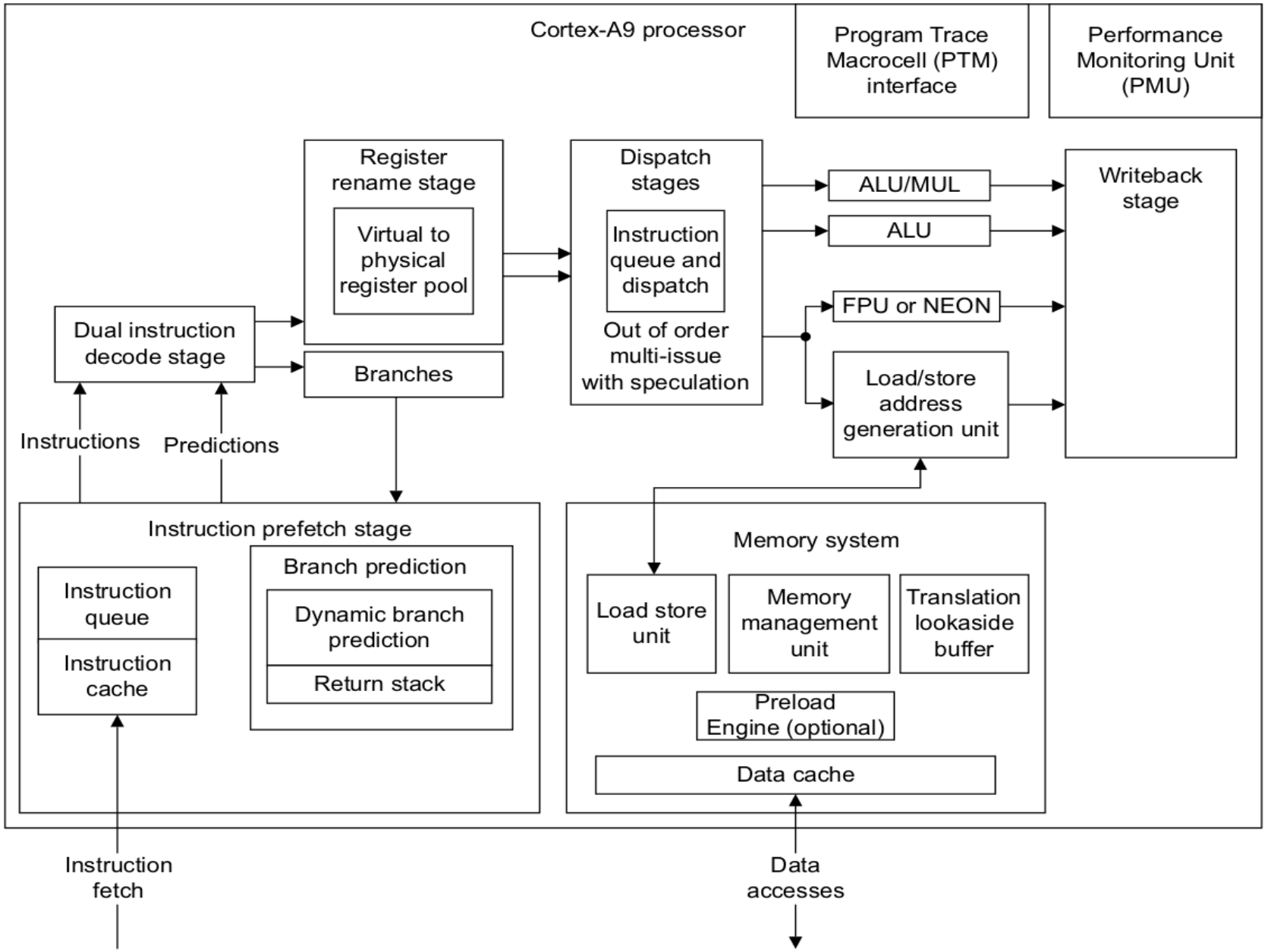
където IPA означава Intermediate Physical Address.

# Виртуализация

*Hypervisor-a се грижи за разпределението на прекъсванията от хардуера към съответната виртуална машина.

*Инструкцията HVC (Hypervisor Call) се използва от софтуера във виртуалната машина, за да извиква функции на hypervisor-a.

# Структурна схема на Cortex-A микропроцесор



# Структурна схема на Cortex-A микропроцесор

Модулите на ковейера са:
*извличане (prefetch)
*декодиране (decode)
*преименуване (register rename)
*диспечериране (dispatch)
*изпълнение (execute – ALU, MUL, FPU, LoadStore)
*обратен запис (write back)
*запис в паметта (memory access)

# Структурна схема на Cortex-A микропроцесор

Някои модули могат да бъдат конфигурирани/включвани/изключвани в IP библиотеката на процесора:

* размер на I-кеш: 16-, 32-, 64-килобайтов
* размер на D-кеш: 16-, 32-, 64-килобайтов
* TLB клетки: 64 / 128
* NEON (векторен FPU): да / не
* FPU (скаларен FPU): да / не
* Интерфейс за трасиране PTM (Program Trace Macrocell) : да / не

# Структурна схема на Cortex-A микропроцесор

*Jazelle инструкции (ускорение за Java приложения): всички / ограничени
* Възможност за power off: да / не
* BTAC (Branch Targer Address Cache): 512 / 1024 / 2048 / 4096
* GHB (Global History Buffer): 1024 / 2048 / 4096 / 8192 / 16384 дескриптора
* Parity check на (RAM, Cache, BTAC+GHB): да / не
* Брой процесори в системата: 1 / 2 / 3 / 4
*други

# Структурна схема на Cortex-A микропроцесор

*В μPU **няма асинхронна логика**.

*Тактовия вход на микропроцесора (μPU) е само един - CLK.

*Тактът на AXI магистралата (AMBA 3 AXI), ACLK, може да бъде **равен** на CLK или **кратен** на него.

*Ако ACLK е кратен, трябва да се използва сигналът ACLKEN0.

*Пример[3]*: на следващият слайд е показан ACLK = CLK / 3.

# Структурна схема на Cortex-A микропроцесор

Figure 2-3 shows a timing example with **ACKLENM0** used with a 3:1 clock ratio between **CLK** and **ACLK** in a Cortex-A9 uniprocessor.

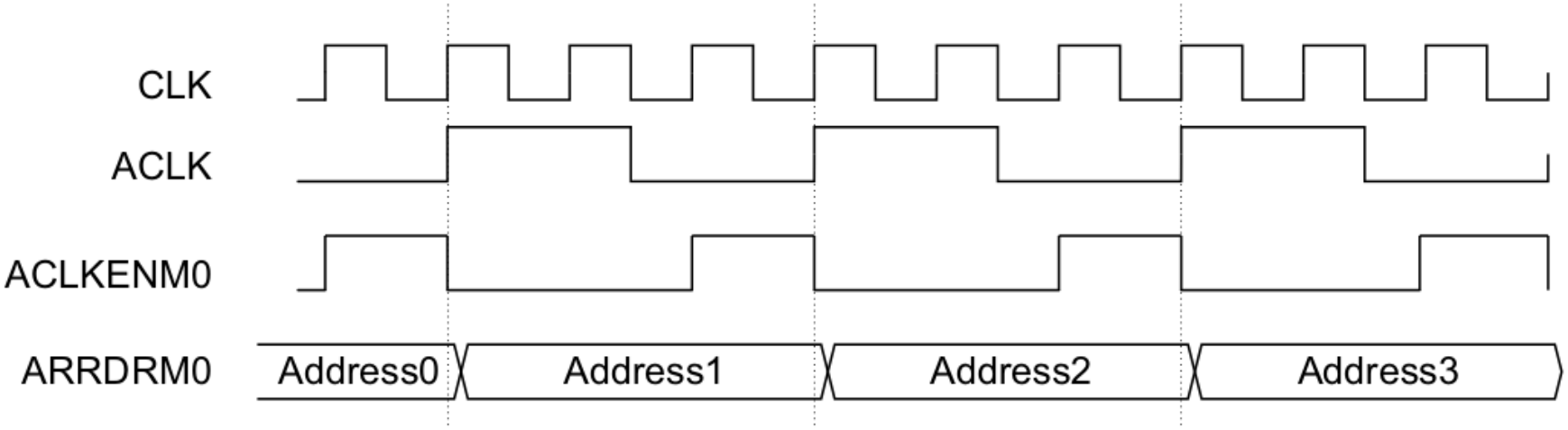



**Figure 2-3 ACLKENM0 used with a 3:1 clock ratio**

The master port, Master0, changes the AXI outputs only on the **CLK** rising edge when **ACLKENM0** is HIGH.

# Структурна схема на Cortex-A микропроцесор

В μPU има 3 региона с отделни захранвания (които могат да бъдат вкл./изкл.):

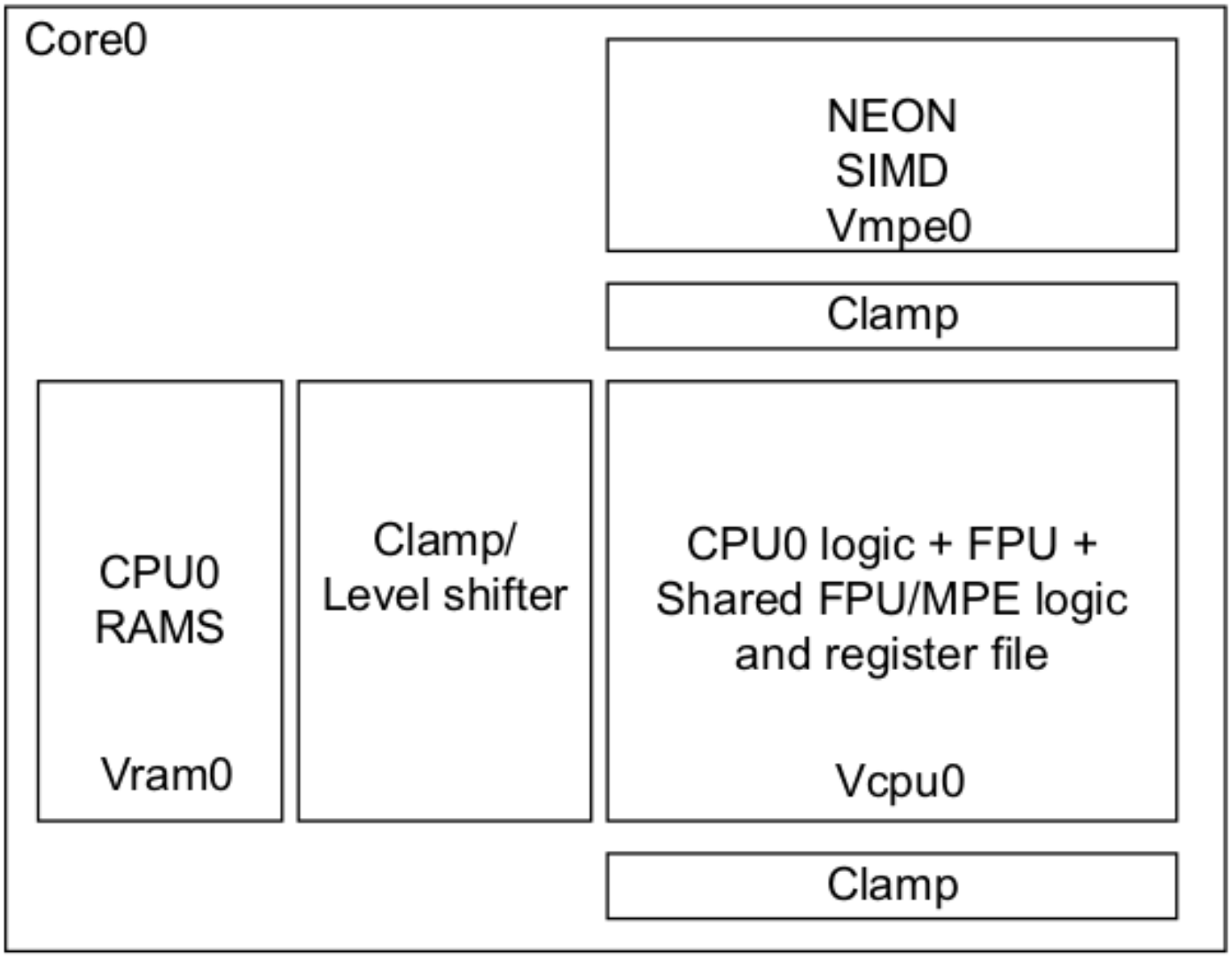
* Cortex-A9 процесорна логика

* Cortex-A9 логика за векторна обработка на данни

* Cortex-A9 RAM памет

# Структурна схема на Cortex-A микропроцесор

Ако се налага да има **изчисления с дроби** и ако те **не използват** масиви (вектори), може да се използва обикновеното FPU, а NEON да се изключи с цел **пестене на енергия.**



# Литература

[1] Arm Architecture Reference Manual for A-profile Architecture, ARM DDI 0487J.a (ID042523), 2023.
[2] ARM Cortex-A Series Programmer’s Guide, Version 4.0, ARM DEN0013D (ID012214), 2013.
[3] Cortex-A9 Technical Reference Manual, r3p0, ARM DDI 0388G (ID072711), 2011.
[4] Learn the architecture – Introducing Neon, Issue 02, 102474_0100_02_en, Arm Limited, 2020.

# Външни връзки

[a] https://www.doc.ic.ac.uk/~eedwards/compsys/float/nan.html
[b] https://www.gnu.org/software/libc/manual/html_node/Infinity-and-NaN.html