Instruções SIMD

Arquitetura AArch64

João Canas Ferreira

Abril 2019

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

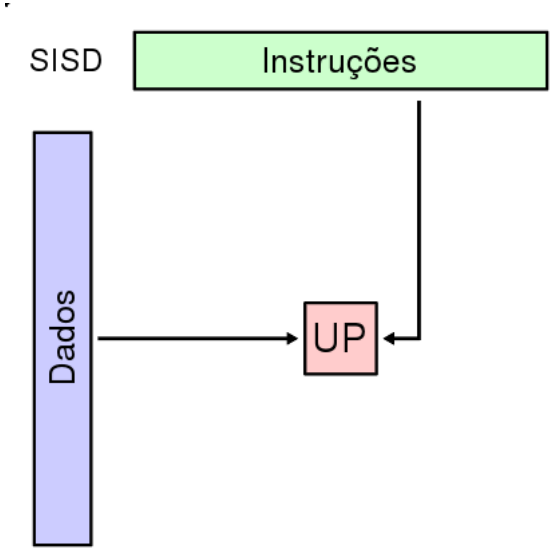
Assuntos

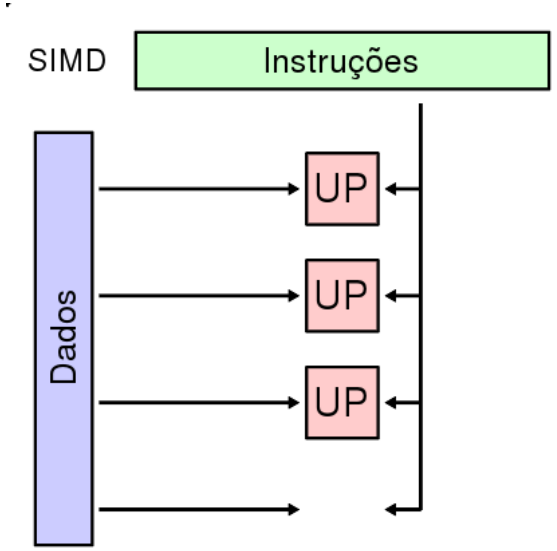
Modelos de execução de instruções

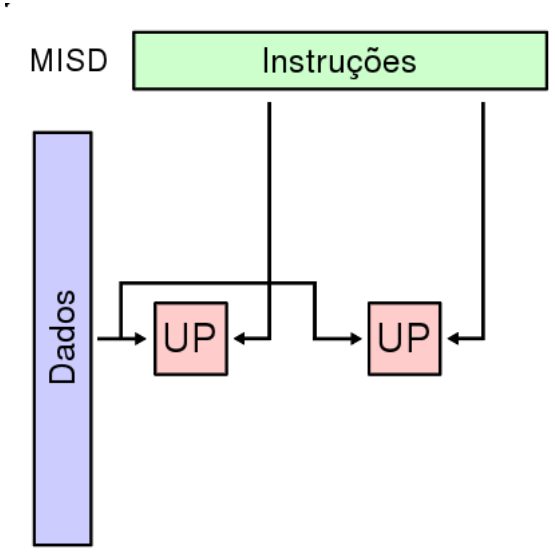
➠ As unidades de processamento podem ser classificadas segundo o número de instruções e de conjuntos de dados tratados em simultâneo:

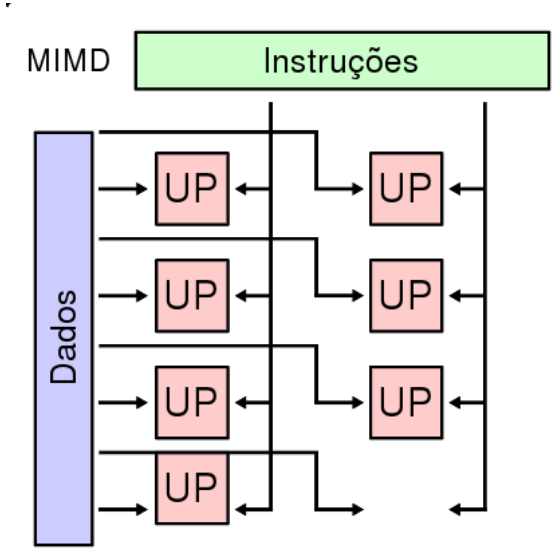
- **SISD:** *Single instruction stream, single data stream*
  processador convencional: 1 instrução processa 1 conjunto de dados
- **SIMD:** *Single instruction stream, multiple data streams*
  1 instrução processa vários conjuntos de dados (processamento vetorial, também usado em GPUs)
- **MISD:** *Multiple instruction streams, single data stream*
  processamento redundante (pouco usado)
- **MIMD:** *Multiple instruction streams, multiple data streams*
  múltiplas instruções (diferentes) processam múltiplos conjuntos de dados (por exemplo, um processador multi-núcleo)

Representação dos modelos de execução



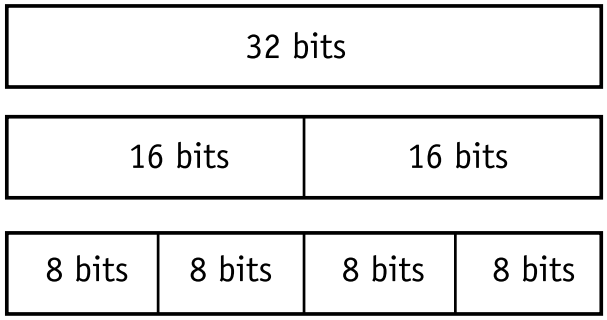


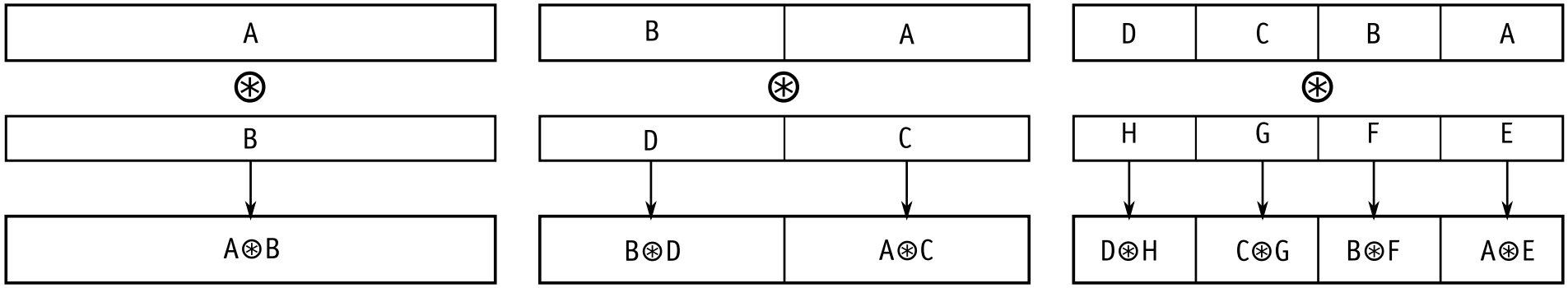




Instruções SIMD: modelo abstrato

➠ Um registo pode ser interpretado como uma unidade ou um vetor com um número fixo (p. ex., 2 ou 4) de registos **independentes**.





➠ Cada elemento do vetor pode ser combinado com o elemento correspondente de outro vetor com uma *única* instrução.

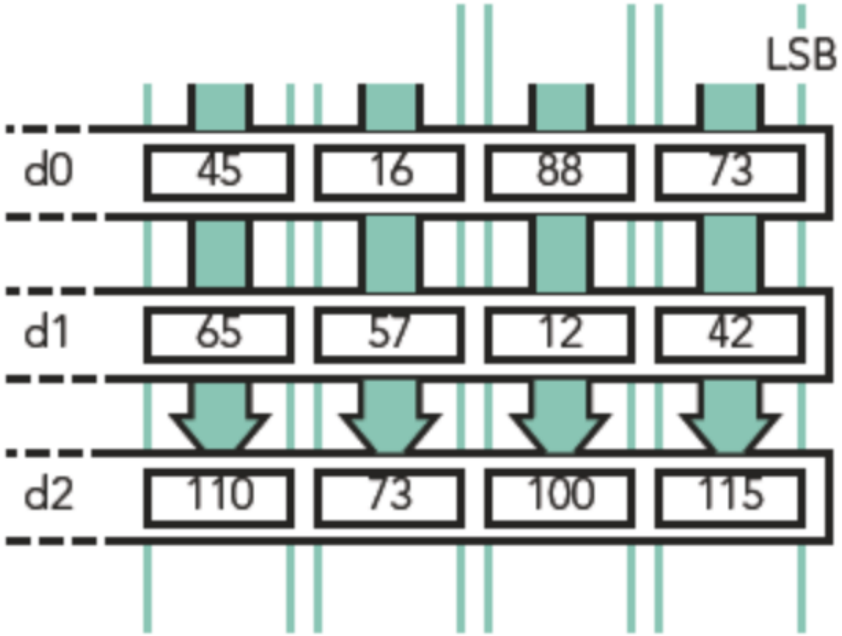
Instruções SIMD: vantagens e desvantagens

- Vantagens
  - Aumento de desempenho
  - Aproveitamento da capacidade de integração (capacidade de integrar número elevado de ALUs e outras unidades de processamento)
  - Paralelismo explícito é mais fácil de aproveitar (operações são naturalmente independentes)
- Desvantagens
  - Requer algoritmos com processamento de dados "uniforme" (todos os elementos do vetor são processados da mesma forma)
  - Necessidade de adaptar a codificação do algoritmo
  - Alguns compiladores têm dificuldade em aproveitar bem este tipo de instruções: recurso a linguagem *assembly*.

Topics

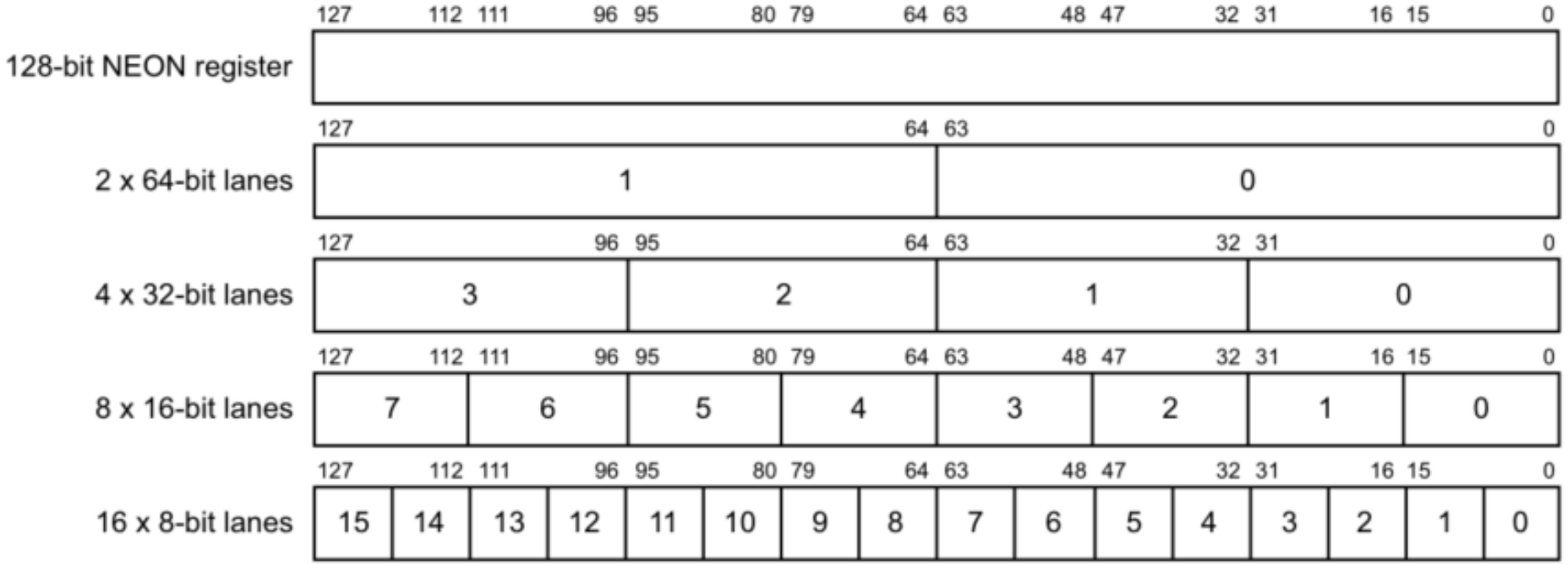
Princípios básicos

- A arquitetura “ARM Advanced SIMD”, as suas implementações e software de apoio são designadas or *tecnologia NEON*.
- Foco: NEON para AArch64
- Suporte para os seguintes tipos de dados:
  - U inteiros sem sinal (*unsigned*) de 8, 16, 32 e 64 bits;
  - S inteiros *com* sinal (cpl 2) 8, 16, 32 e 64 bits;
  - F números em vírgula flutuante de 32 e 64 bits.
- Na terminologia ARM, os cálculos SIMD são realizador por faixa (*lane*)



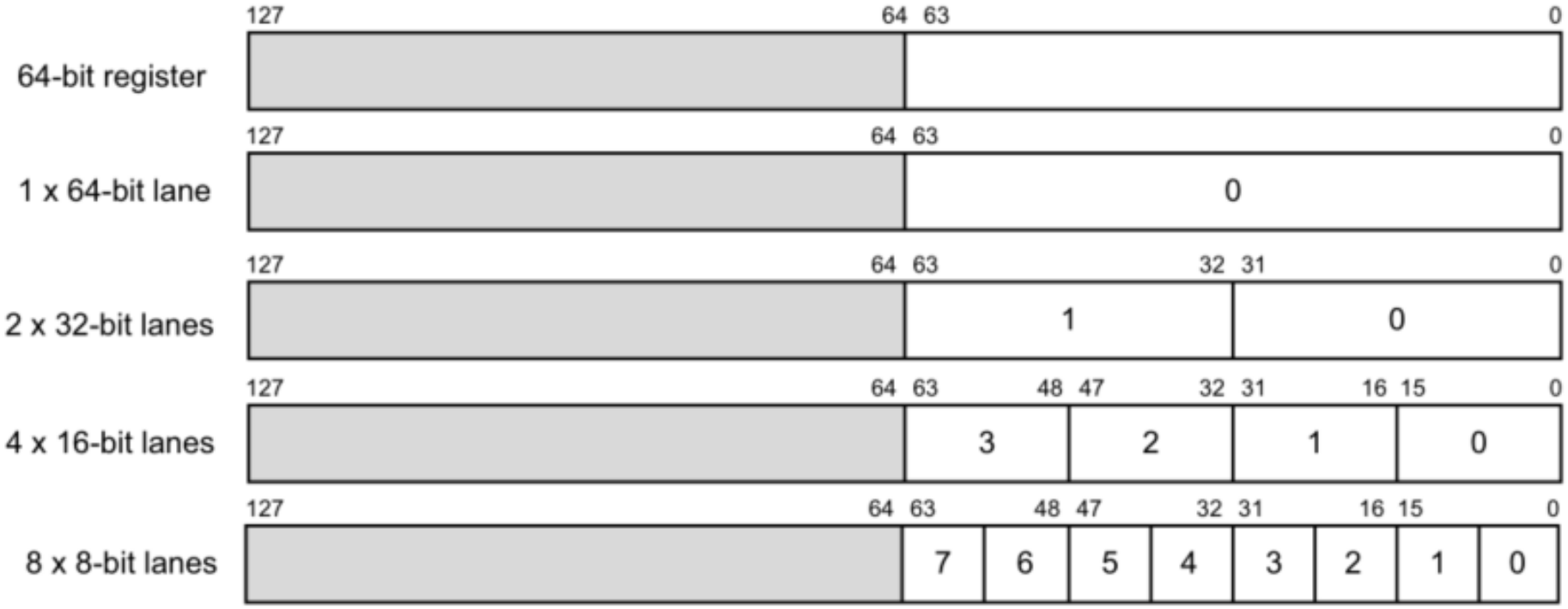
Banco de registo NEON

- O banco de registos tem 32 registos de 128 bits (`quadword`): `V0-V31`.
- É *o mesmo banco de registos* que é usado para os operandos de vírgula flutuante.



Banco de registo NEON (2)

➠ O banco de registos também pode ser considerado como contendo 32 registos de 64 bits (`doubleword`): `D0-D31`.



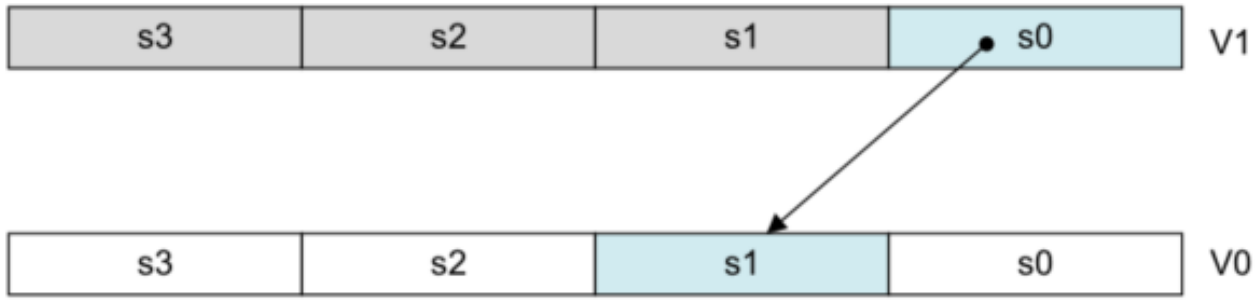
Valores escalares e vetores

- Os registos V<n> podem ser considerados como pequenos vetores (1, 2, 3, 4, 8 ou 16 componentes).
- Geralmente, as instruções SIMD operam sobre vetores, mas algumas instruções usam um componente apenas.

```
<Instrução> Vd.TS[i], Vn.Ts[j]
```

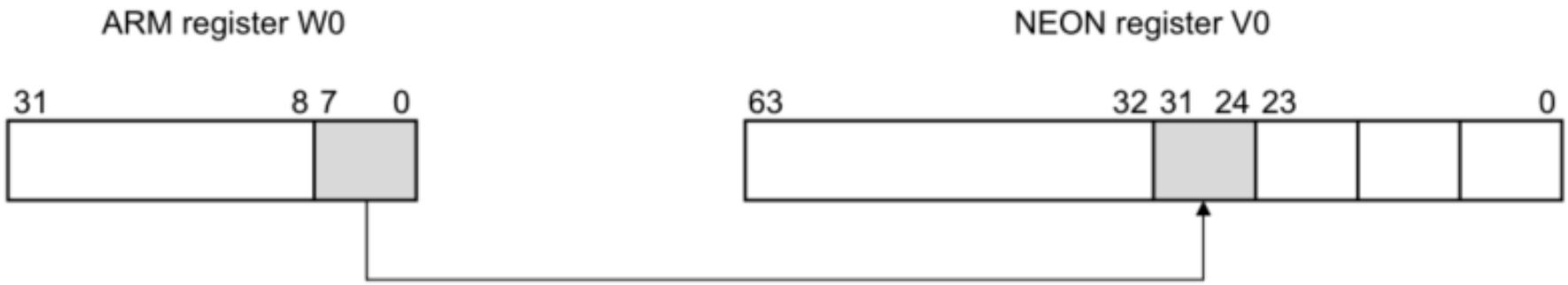
Vd	registo de destino
Vn	registo de origem
Ts	especificador de tipo
i, j	índices dos elementos

- Exemplo: `INS V0.S[1], V1.S[0]`



Valores escalares

- Valores escalares podem ter 8, 16, 32 ou 64 bits
- Instruções que usam valores escalares pode aceder a qualquer elemento do banco de registos
- Exemplo: `MOV V0.B[3], W0`



- Exceção: *instruções de multiplicação* só usam escalares de 16 ou 32 bits e só podem aceder aos primeiros 128 escalares do banco de registos
  - escalares de 16 bits: Vn.H[x], com $0 \leq n \leq 15$, $0 \leq x \leq 7$
  - escalares de 32 bits: Vn.S[x], com $0 \leq n \leq 15$, $0 \leq x \leq 3$

Topics

Mnemónicas têm utilização expandida

➠ A mesma mnemónica pode representar várias instruções (i.e., resultam em codificações diferentes).

➠ Exemplo:

- `ADD W0,W1,W2` instrução básica (A64)
- `ADD V0.4H,V1.4H,V2.4H` instrução vetorial (NEON)

➠ Adicionar à mnemónica um dos prefixos S, U ou F para indicar o tipo. (Se fizer sentido.)

- `FADD D0, D1, D2`

➠ A organização do vetor (tamanho do elemento e número de "faixas") é definido pelo qualificador do registo: 8B, 16B, 4H, 8H, 2S, 4S, 2D.

Exemplo: realizar duas adições simultâneas de valores de 64 bits

- `ADD V0.2D,V1.2D,V2.2D` inteiros com sinal
- `FADD V0.2D,V1.2D,V2.2D` VF, precisão dupla

➠ Não existem instruções para cálculo com números VF de meia precisão!

Variantes de instruções NEON

➠ Algumas instruções NEON estão disponíveis nas variantes *Normal*, *Long*, *Wide*, *Narrow* ou *Saturating*.

As variantes *Long*, *Wide* e *Narrow* são indicadas por um sufixo no nome: **L**, **W** e **N**, respetivamente.

A variante *Saturating* usa um dos prefixos **SQ** ou **UQ** consoante se trate de operandos com ou sem sinal, respetivamente.

➠ A variante *Normal* opera sobre qualquer tipo de vetor e produz vetores da mesma dimensão (número de componentes) e (geralmente) do mesmo tipo.

➠ Para além das variantes, existem também instruções que operam sobre pares de registos adjacentes (operandos `doubleword` ou `quadword`).

Estas instruções usam o sufixo **P**.

Variante "long"

- Operam sobre vectores de 64 bits e produzem um vetor de 128 bits.
- Os elementos do resultados têm o dobro do tamanho dos operandos.
- Exemplo: `SADDL V0.4S,V1.4H,V2.4H`