Procesadores Vectoriales

Un procesador vectorial proporciona instrucciones que trabajan sobre vectores.
Una sola instrucción de vectores es equivalente a un lazo completo en el que se procesa un elemento a la vez.
Esto ahorra:

- El uso y actualización de índices
  - De control del bucle.
  - De indexación al elemento adecuado en cada iteración.
- Los saltos tras cada iteración al comienzo del bucle
- La búsqueda (fetch) de todas las instrucciones del bucle

Propiedades de las operaciones con vectores que propician el paralelismo:

- Cada resultado es independiente de los resultados previos
  - No hay riesgos RAW
- Los accesos a memoria tienen un patrón conocido porque:
  - Se explota el principio de localidad
    - Los elementos están en lugares contiguos, o
    - En lugares a distancia uniforme
- Como sustituyen a bucles, los riesgos de control disminuyen al no existir saltos

Arquitectura Vectorial básica

Un procesador vectorial consta, típicamente, de:

- Una Unidad Escalar segmentada ordinaria
- Una Unidad Vectorial.
  - Todas las Unidades funcionales dentro de la unidad vectorial tienen una latencia de varios ciclos de reloj.

Tipos de arquitecturas vectoriales

- Procesadores *Registro-Vector (RV)*
  - Todas las operaciones ALU son entre registros (registro-vector)
  - Operaciones de carga y almacenamiento trasladan datos entre memoria y registro-vector

Desde los años 90, son las más importantes

- Ejemplos: Cray Research: CRAY-1, CRAY-2, X-MP, Y-MP

japoneses: NEC SX/2 Fujistu VP200

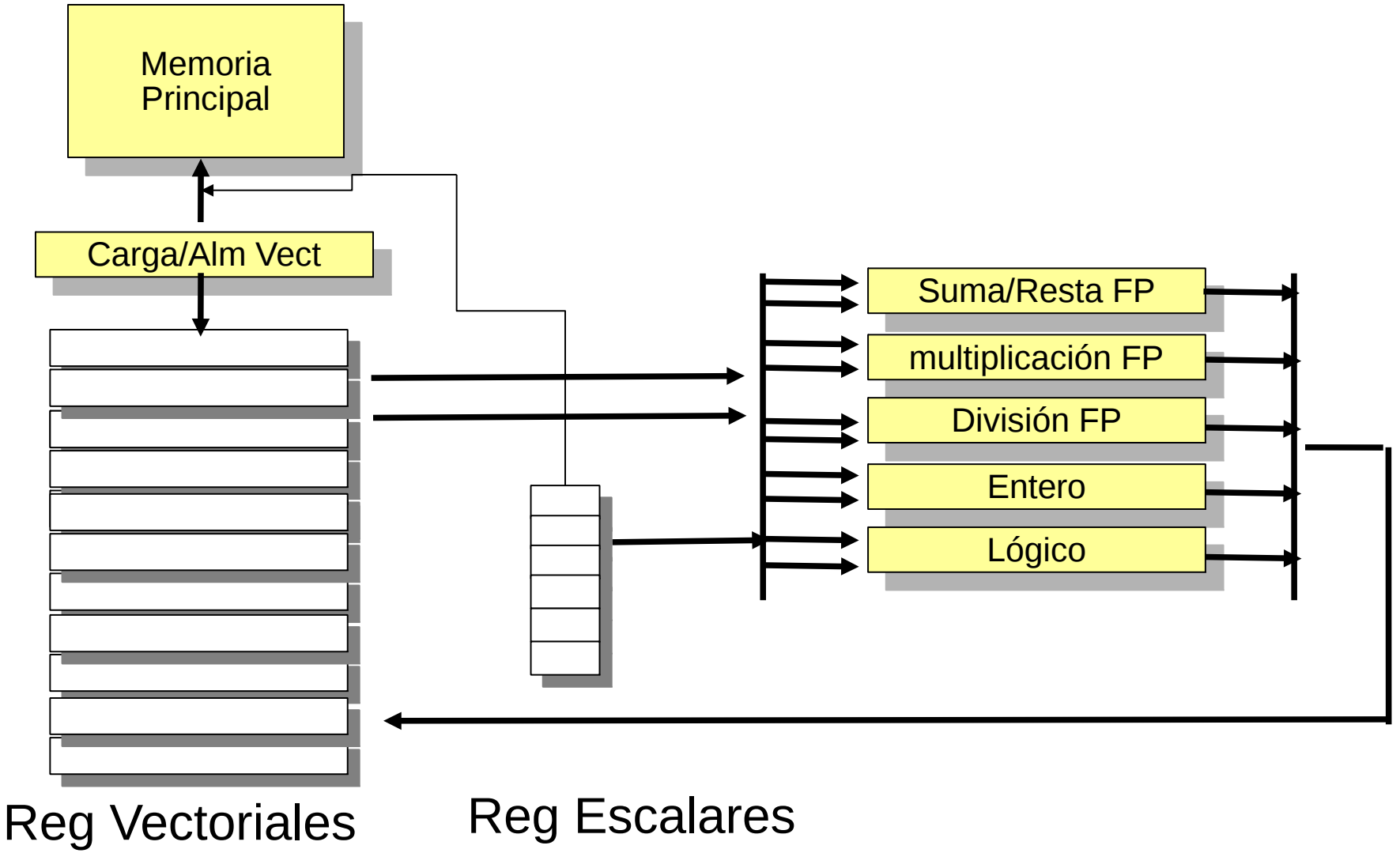
- Procesadores V*ector Memoria-Memoria*

Todas las operaciones de vectores son entre memoria y memoria.

Los primeros computadores vectoriales fueron así

- Ej: los CDC´s

Estructura básica vectorial RV



Características DLXV

- Extensión de DLX para P. Vectorial
- No usa CACHE sino MP
  - Acceso simultaneo a memoria con escalares
    - escalares a MCD y vectoriales a MP
  - Unidades carga-almacenamiento segmentadas
    - tras latencia inicial se transfiere una componente por ciclo de reloj
- Registros especiales:
  - De longitud vectorial: vlr
  - De máscaras vectorial: vm

Instrucciones vectoriales: DLXV

- addv V1,V2,V3 V1 <-- V2 + V3 suma cada elemento
- addsv V1, F0, V2 V1 <-- V2 + F0 suma F0 a cada elemento
- lv V0, n(r1) V0 ← M(n+r1) carga tantos como vlr
- movi2s vlr, r1 vlr ← r1 longitud del vector
- sv n(r2), v1 M(n+r2) ← v1 almacena V1 desde
- lvws v1,r1,r2 Load Vector v1 from M(r1) With Separation r2
- svws r1,r2,v1 Store...

Instrucciones vectoriales: DLXV

- CVI V1, R1 Create Vector Index V1 con los valores:
  - 0*R1 en V1(0),
  - 1*R1 en V1(1),
  - 2*R1 en V1(2),
  - 3*R1 en V1(3)... un vector de índices
- SVI R3,V1, V2 M(R3+V1) <-- V2 :
  - M(R3+V1(0)) ← V2(0)
  - M(R3+V1(1)) ← V2(1)
  - M(R3+V1(2)) ← V2(2) . V1 es un índice, R3 dir comienzo

```
; Ejemplo escalar-vectorial
;
; Bucle DAXPY con instrucciones vectoriales:
; Realiza la operacion Z = cte*X + Y donde X e Y son vectores
; y cte es un escalar.
;
        .data
x:       .double 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
        .double 10, 11, 12, 13, 14, 15, 16, 17, 18, 19
        .double 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
        .double 30, 31, 32, 33, 34, 35, 36, 37, 38, 39
        .double 40, 41, 42, 43, 44, 45, 46, 47, 48, 49
        .double 50, 51, 52, 53, 54, 55, 56, 57, 58, 59
        .double 60, 61, 62, 63

y:       .double 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
        .double 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
        .double 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
        .double 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
        .double 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
        .double 100, 100, 100, 100, 100, 100, 100, 100, 100, 100
        .double 100, 100, 100, 100

z:       .space 64*8

cte:       .double 3,1415
```

```
; Código vectorial DLX ISA

main:
        addi    r4,r0,64        ; 64 elementos

        ld      f0, cte(r0)     ; f0 escalar
        movi2s  vlr,r4          ; vlr : longitud de los vectores

        lv      v0,x(r0)
        multsv  v1,f0,v0        ; multsv  multiply scalar * vector

        lv      v2,y(r0)
        addv    v3,v1,v2        ; v3 ← v1 + v2  add vectores

        sv      z(r0),v3        ; store vector
```

```
; Código en máquina escalar (convencional)  DLX ISA

ini:    addi   r1,r0,r0
        ld     f0,cte(r0)          ; f0 escalar

        addi   r5,r0,64            ;64 elementos
loop:
        ld     f2,x(r1)
        multd  f2,f0,f2
        ld     f4,y(r1)
        addd   f4,f2,f4
        sd     z(r1),f4

        addi   r1,r1,8
        subi   r5,r5,1
        bnez   r5,loop
```