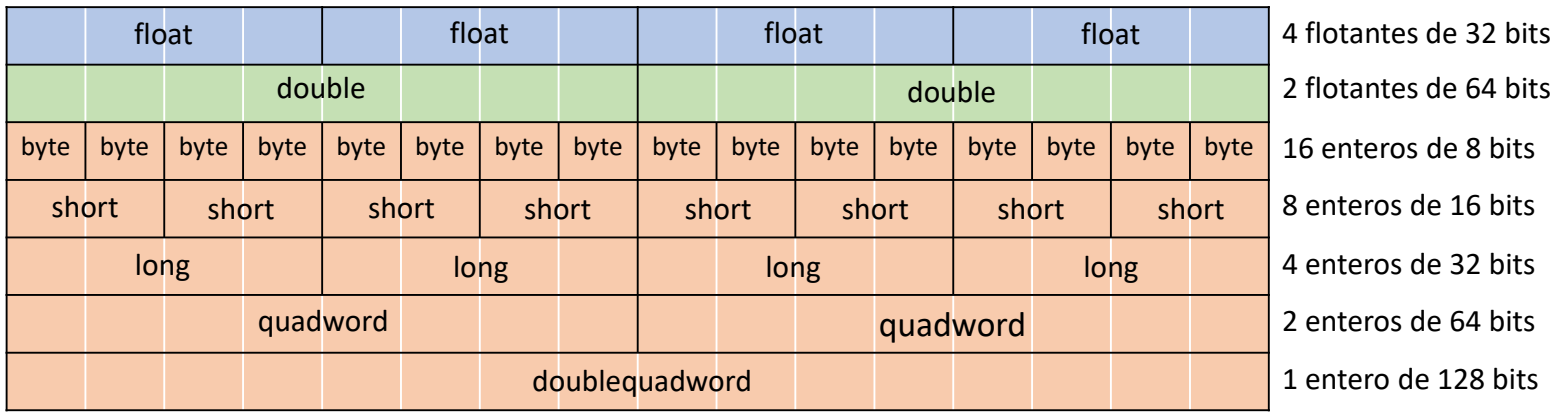
Instrucciones de punto flotante

Arquitectura del Computador - LCC - FCEIA-UNR

8 de octubre de 2022

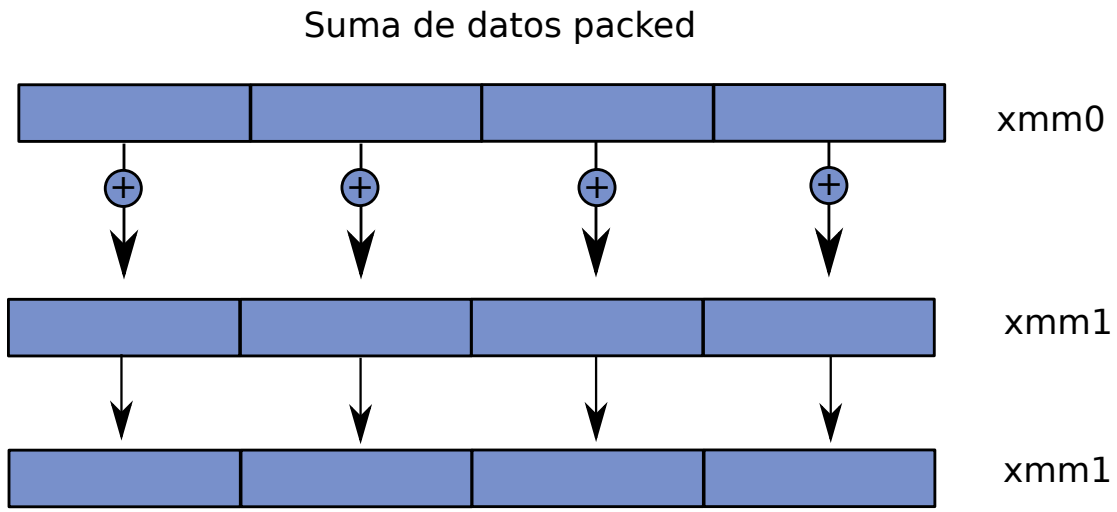
Instrucciones de punto flotante

- Las operaciones de punto flotante se realizaban en un chip separado (8087).
- Las CPU x86-64 tienen 16 registros de punto flotante (128 bits): XMM0-XMM15.



Instrucciones de punto flotante

- Estos registros se pueden utilizar para instrucciones de datos individuales o instrucción instrucciones de datos múltiples (SIMD, *Single Instruction Multiple Data*).



Mover escalares hacia o desde registros de punto flotante

- movss mueve un valor en punto flotante simple precisión 32 bits a/desde un registro XMM.
- movsd mueve un valor en punto flotante doble precisión 64 bits a/desde un registro XMM.
- Las instrucciones siguen el patrón estándar de tener posiblemente una dirección de memoria:

```
movss x, %xmm0          # mueve el valor en x en xmm0
movsd %xmm1, y          # mueve el valor de xmm1 a y
movss %xmm0, %xmm2      # mueve de xmm0 a xmm2
```

Moviendo datos empaquetados

- Los registros XMM son de 128 bits
- Pueden contener 4 flotantes o 2 dobles (o enteros de varios tamaños)
- En las CPU más nuevas se amplían a 256 bits y se denominan YMM.
- movaps mueve 4 flotantes hacia/desde una dirección de memoria alineada a 16 bytes.
- movups hace la misma tarea con direcciones de memoria no alineadas.
- movapd mueve 2 dobles hacia/desde una dirección de memoria alineada a 16 bytes.
- movupd hace la misma tarea con direcciones de memoria no alineadas.

```
movups x, %xmm0      # mueve 4 flotantes a xmm0
movupd %xmm15, y     # mueve 2 dobles a y
```

Suma en punto flotante

- Hay 2 operandos: destino y origen.
- El operando fuente puede ser memoria o un registro XMM
- El operando destino debe ser un registro XMM
- Las banderas no se ven afectadas.
- addss suma un flotante escalar (precisión simple) a otro.
- addsd suma un flotante escalar (doble simple) a otro.
- addps suma 4 flotantes a 4 flotantes - suma por pares
- addpd suma 2 dobles a 2 dobles

```
movss a, %xmm0    # carga a
addss b, %xmm0    # suma b + a
movss %xmm0, c    # guarda la suma en c
movapd a, %xmm0   # carga 2 dobles desde a
addpd b, %xmm0    # suma a[0]+b[0] y a[1]+b[1]
movapd %xmm0, c   # guarda las dos sumas en c
```

Conversión a un punto flotante de longitud diferente

- cvtss2sd convierte de float a double (escalares).
- cvtps2pd convierte 2 floats empaquetados a 2 doubles empaquetados.
- cvtsd2ss convierte escalares doubles a escalares floats.
- cvtpd2ps convierte 2 doubles empaquetados a 2 floats empaquetados.

```
cvtss2sd x, %xmm0          # carga x en xmm0 como double
addsd y, %xmm0             # suma un double a x
cvtsd2ss %xmm0, %xmm0      # convierte a float
movss %xmm0, c             # guarda en c el resultado
```

Convirtiendo punto flotante a/desde entero

- cvtss2si convierte un float a un double word o quad word entero.
- cvtsd2si convierte un float a un double word o quad word entero.
- Estas dos instrucciones redondean el valor.

- cvttss2si y cvttsd2si convierten por truncado.
- cvtsi2ss convierte un entero a un float en un registro XMM.
- cvtsi2sd convierte un entero a double en un registro XMM.

```
cvtss2si %xmm0, %eax    # convierte a entero double word
cvtsi2sd %rax, %xmm0    # convierte quad word a double
cvtsi2sdq x, %xmm0      # convierte double word entero a
                        # double
```

Comparaciones

- ucomiss compara floats.
- ucomisd compara doubles.
- El segundo operando debe ser un registro XMM.
- Se setean las banderas zero, parity y carry (las restantes permanecen nulas).

```
        movss a, %xmm0
        mulss b, %xmm0
        ucomiss c, %xmm0
        jbe menor_igual     # salta si a*b <= c
        .....
menor_igual:
        .....
```