Université KASDI MERBAH – Ouargla
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique et Technologie de l'information

Cours architectures des ordinateurs

Cours 4:
Le FPU (Float Point Unit)

Enseignante: Chafika Benkherourou

Plan

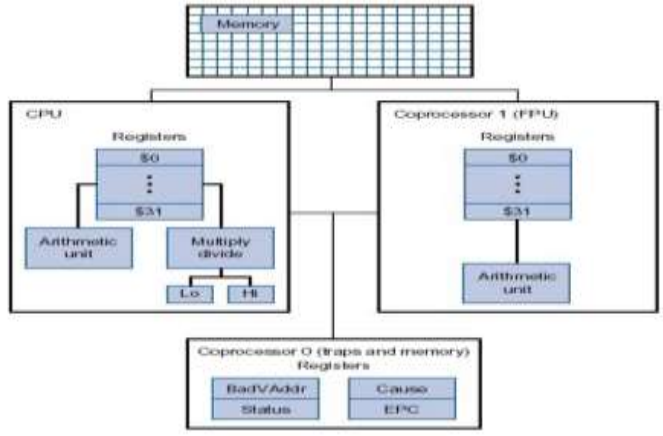
- Introduction
- Le FPU
- Représentation des nombres réels
- Opérations de transfert sur les nombres réels
- Les instructions de conversion Entier/Réel
- Opérations arithmétiques
- Exemple
- Les codes des appels système

Introduction

- La gestion de la virgule flottante est complexe par rapport aux nombres entiers.
- Dans MIPS R3000, on utilise un processeur spécialement conçu pour les calculs en virgule flottante.
- C'est le FPU (Float Point Unit)

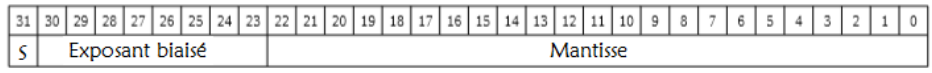
Float Point Unit: (FPU)

- Il est chargé d'effectuer les opérations arithmétiques à virgules flottantes.
- On l'appelle aussi le Co-processeur 1.
- Il possède 32 registres nommés: $f0 – $f31

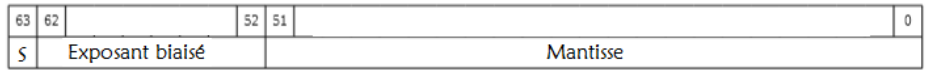


Représentation des nombres réels:

- Il existe des normes pour représenter les nombres réels. La plus employée actuellement est la norme IEEE 754.
- Le format d'un nombre en virgule flottante est:



- S: le signe (0=> Positif, 1=>négatif)
- La représentation simple précision utilise 32 bits (ex: le registre $f6)
- La représentation double précision utilise 64 bit (soit deux registres successifs, ex: les registre $f11 et $f12)



Représentation des nombres réels:

❑ Exemple: Représentation du nombre $4{,}625_{10}$ en simple précision.

- Etape 1: Conversion du nombre 4,625 en binaire ($4{,}625_{10} = 100{,}101_2$)
- Etape 2: Normalisation scientifique. La partie entière est égale à 1
  ($100{,}101 = 1{,}00101 \times 2^2$)
- Etape 3: Calcul de l'exposant biaisé. Le biais pour la norme IEEE 754 simple précision est 127.
  - L'exposant biaisé = Exposant réel + biais = 2 + 127 =129 = 10000001_2

Opérations sur les nombres réels:

- Les opérations en virgule flottante sont similaires aux opérations sur les nombres entiers. Sauf qu'elles utilisent les registres à virgule flottante ($f0-$f31).

- Les variables en virgule flottante sont déclarées par:

  - Float : pour la simple précision
  - Double: pour la double précision

Exemple:

```
pi:     .float      3.14159
tao:    .double     6.28318
```


Opérations de transfert sur les nombres réels:

- l.s $rd, mem (transfert de la variable mem en mémoire vers le registre $rd)
- l.d $rd, mem
- s.s $rs, mem (copie le contenu de $rs dans la variable mem en mémoire)
- s.d $rs, mem

- mov.s $rd,$rs
- mov.d $rd,$rs

Note:

- **s:** pour simple précision
- **d:** pour double précision

```
fnum1:      .float      3.14
fnum2:      .float      0.0
dnum1:      .double     6.28
dnum2:      .double     0.0
```

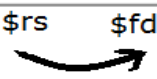
```
l.s     $f6, fnum1
s.s     $f6, fnum2

l.d     $f6, dnum1
mov.d   $f8, $f6

s.d     $f8, dnum2
```

Opérations de transfert sur les nombres réels:

- Les opérations arithmétiques utilisent les nombres réels et les nombres entiers. Pour le transfert de données entre les registres réels et les registres entiers on utilise des instructions spéciales:

Instruction		Description
`mfc1`	$rd $fs 	Copie le contenu du registre $fs (du co-processeur 1) dans le registre $rd (Entier)
`mfc1.d`	$rd $fs 	Copie l contenu des registres $fs et $fs+1 (du co-processur1) dans les registres $rd et $rd+1 (Entier)
`mtc1`	$rs $fd 	Copie le contenu du registre $rs dans le registre $fd (co-processur1)
`mtc1.d`	$rs $fd 	Copie le contenu des registres $rs et $rs+1 dans les registres $fd et $fd+1 (co-processeur)

Opérations de transfert sur les nombres réels:

- **Exemple:**
  - Pour copier le contenu du registre ***$f12*** dans le registre entier ***$t1***, on utilise l'instruction suivante:

```
mfc1        $t1, $f12
```


  - Pour copier le contenu du registre ***$s0*** dans le registre entier ***$f12***, on utilise l'instruction suivante:

```
mtc1        $s0, $f12
```


  - **Note:**
    - Les valeurs transférées doivent être converties.

Les instructions de conversion Entier/ Réels:

- Lorsque les données sont transférées entre les registres entiers et les registres à virgule flottante, une conversion est nécessaire:

Instruction	Description
`cvt.d.s FRdest, FRsrc`	Convertit la valeur réelle 32-bits contenue dans le registre FRsrc en double précision et la mettre dans le registre destination FRdest
`cvt.d.w FRdest, FRsrc`	Convertit la valeur entière 32-bits en double précision et la mettre dans le registre FR dest
`cvt.s.d FRdest, FRsrc`	Convertit la valeur réelle 64-bits en simple précision
`cvt.s.w FRdest, FRsrc`	Convertit la valeur entière 32-bits en simple précision
`cvt.w.d FRdest, FRsrc`	Convertit la valeur réelle 64-bits en entier 32-bits
`cvt.w.s FRdest, FRsrc`	Convertit la valeur réelle 32-bits en valeur entière 32-bits

Les instructions de conversion Entier/ Réels:

- Exemple:
  - On donne la déclaration des données suivante:

```
iNum:       .word       42
fNum:       .float      0.0
```

  - On veut convertir la valeur entière contenue ***iNum*** en valeur réelle simple précision et la mettre dans ***fNum***:

```
lw          $t0, iNum
mtc1        $t0, $f6
cvt.s.w     $f8, $f6
s.s         $f8, fNum
```

    - La valeur entière ***iNum*** est transférée dans le registre ***$t0***.
    - Le contenu de ***$t0*** est copié dans le registre ***$f6*** dans le Co-processeur1 par la commande ***mtc1***.
    - Une conversion en simple précision est réalisée avec la commande ***cvt.s.w*** et la valeur est copiée dans ***$f8***.
    - Le contenu du registre ***$f8*** est enregistré dans la variable réelle ***fNum***.

Les instructions de conversion Entier/ Réels:

- Exemple:
  - On donne la déclaration des données suivante:

```
pi:         .double     3.14
intPi:      .word       0
```

  - On veut convertir la valeur réelle 64-bits contenue ***Pi*** en valeur entière et la mettre dans ***intPi:***

```
l.d         $f10, pi
cvt.w.d     $f12, $f10
mfc1        $t1, $f12
sw          $t1, intPi
```

    - La valeur réelle double ***Pi*** est transférée dans le registre ***$f10***.
    - Une conversion en entier est réalisée avec la commande ***cvt.w.d*** et la valeur est copiée dans ***$f12.***
    - Le contenu de ***$t12*** est copié dans le registre ***$t1*** dans le Co-processeur1 par la commande ***mfc1***.
    - Le contenu du registre ***$t1*** est enregistré dans la variable réelle ***intPi.***

Opérations arithmétiques sur les nombres réels:

- Les tableau suivant donne les opérations arithmétiques de base:

Instruction	Description
`add<type> FRdest, FRsrc, FRsrc`	FRdest = FRsrc + FRsrc
`sub<type> FRdest, FRsrc, FRsrc`	FRdest = FRsrc - FRsrc
`mul<type> FRdest, FRsrc, FRsrc`	FRdest = FRsrc * FRsrc
`div<type> FRdest, FRsrc, FRsrc`	FRdest = FRsrc / FRsrc
`rem<type> FRdest, FRsrc, FRsrc`	FRdest = FRsrc % FRsrc

- Note:

```
add<type> FRdest, FRsrc, FRsrc
```

  - <type> : définit le type de la précision simple (.s) ou double (.d)

Opérations arithmétiques sur les nombres réels:

- Exemple:
  - On donne la déclaration des données suivante:

```
fnum1:      .float      6.28318
fnum2:      .float      3.14159
fans1:      .float      0.0
fans2:      .float      0.0

dnum1:      .double     42.3
dnum2:      .double     73.6
dans1:      .double     0.0
dans2:      .double     0.0
```

  - On veut réaliser les opérations suivantes:

```
fans1 = fnum1 + fnum2
fans2 = fnum1 * fnum2
dans1 = dnum1 - dnum2
dans2 = dnum1 / dnum2
```

Opérations arithmétiques sur les nombres réels:

- Exemple:

```
l.s       $f4, fnum1
l.s       $f6, fnum2
add.s     $f8, $f4, $f6
s.s       $f8, fans1          # fans1 = fnum1 + fnum2
mul.s     $f10, $f4, $f6
s.s       $f10, fans2         # fans2 = fnum1 * fnum2

l.d       $f4, dnum1
l.d       $f6, dnum2
sub.d     $f8, $f4, $f6
s.d       $f8, dans1          # dans1 = dnum1 - dnum2

div.d     $f10, $f4, $f6
s.d       $f10, dans2         # dans2 = dnum1 / dnum2
```

- Notes:
  - Dans l'instruction **l.d $f4, dnum1**, les registres **$f4** et **$f5** sont utilisés.

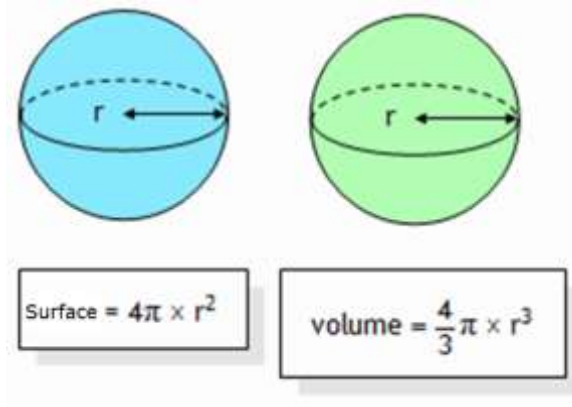
Les codes des appels système:

System Call Codes

Service	Code (put in $v0)	Arguments	Result
print_int	1	$a0=integer	
print_float	2	$f12=float	
print_double	3	$f12=double	
print_string	4	$a0=addr. of string	
read_int	5		int in $v0
read_float	6		float in $f0
read_double	7		double in $f0
read_string	8	$a0=buffer, $a1=length	
sbrk	9	$a0=amount	addr in $v0
exit	10		

Exemple:

- Calculer la surface et le volume d'une sphère:



Exemple:

- Solution:

```
#  Data Declarations

.data

pi: .float 3.14
fourPtZero: .float 4.0
threePtZero: .float 3.0
rayon: .float 17.25

surfaceArea: .float 0.0
volume: .float 0.0
msg0: .asciiz "Entrez le rayon: "
msg1: .asciiz "Surface = "
msg2: .asciiz "Volume = "
ret: .asciiz "\n"

# ---------------------------------
.text
main:
```

Exemple:

```
    l.s $f2,fourPtZero
    l.s $f4,pi
    mul.s $f4,$f2,$f4    # 4.0 * pi

    la $a0,msg0
    li $v0,4
    syscall

    li $v0,6    # Number to $f0
    syscall

    mov.s $f6,$f0
    # ----- #  Calculate surface area of a sphere.
    mul.s $f8,$f6,$f6    # rayon^2
    mul.s $f8,$f4,$f8    # 4.0 * pi * rayon^2
    s.s $f8,surfaceArea  # store final answer
    mov.s $f12,$f8
```

Exemple:

```
la $a0,msg1
li $v0,4
syscall

li $v0,2
syscall

la $a0,ret
li $v0,4
syscall

# ----- #  Calculate volume of a sphere.

l.s $f8,threePtZero
div.s $f5,$f4,$f8    # (4.0 * pi / 3.0)
mul.s $f10,$f6,$f6
mul.s $f10,$f10,$f6  # rayon^3
mul.s $f12,$f5,$f10  # 4.0*pi/3.0*rayon^3
s.s $f12,volume      # store final answer
```


Exemple:

```
la $a0,msg1
li $v0,4
syscall

li $v0,2
syscall

la $a0,ret
li $v0,4
syscall

# ----- #  Calculate volume of a sphere.

l.s $f8,threePtZero
div.s $f5,$f4,$f8    # (4.0 * pi / 3.0)
mul.s $f10,$f6,$f6
mul.s $f10,$f10,$f6  # rayon^3
mul.s $f12,$f5,$f10  # 4.0*pi/3.0*rayon^3
s.s $f12,volume      # store final answer
```

Exemple:

```
la $a0,msg2
li $v0,4
syscall

li $v0,2
syscall

# ----- #  Done, terminate program.

li $v0,10
syscall

.end main
```