Pràctica 2
Aritmètica d'enters (1), operacions lògiques i Entrada/Eixida

2.1 Objectius

- Prendre contacte en algunes funcions del sistema d'entrada i sortida
- Conèixer instruccions aritmètiques d'enters
- Conèixer les instruccions lògiques
- Introduir la codificació d'instruccions

2.2 Material

Simulador MARS i un codi font de partida

2.3 Desenvolupament pràctic

2.3.1 *Input* i *output*

El simulador del MIPS conté un sistema operatiu rudimentari que permet als programes demanar operacions d'E/E. D'aquesta manera podem interactuar amb l'usuari fent lectures de dades des del teclat o imprimint valors en la pantalla. Això es farà fent servir la instrucció `syscall`. (vegeu en la finestra `Mars Help> MIPS>Syscalls`). La instrucció *syscall* ens permet, doncs, cridar els serveis del sistema que ens donaran accés als dispositius d'entrada i d'eixida com pot ser la consola de l'usuari, el teclat i altres dispositius externs. El valor que hi haja en el registre *$v0* ens permet seleccionar una entre totes les funcions dels sistemes d'entrada i eixida. En alguns casos caldrà passar paràmetres o rebre'ls del sistema d'entrada i eixida; en aquests casos es faran servir determinats registres. Per tant, la seqüència d'invocació a una funció del sistema serà col·locar en *$v0* l'identificatiu de la funció i a continuació

fer la crida de la instrucció *syscall*. A continuació es proposen diferents exemples perquè observeu la utilització de les funcions d'entrada i eixida per la consola i conegueu els efectes.

2.3.1.1 Activitat 1

Imprimir en la consola un valor enter. S'utilitzarà la **funció 1** (`print integer`). Si fiquem un 1 en el registre *$v0* s'escriu el valor de l'enter contingut en $a0 en la consola Run I/O.

```
#################################
#                               #
#     Codi de l'activitat 1     #
#     Imprimir en consola       #
#                               #
#################################

.text 0x0040000
addi $a0,$0,25    # Valor a escriure en $a0
addi $v0,$0,1     # Funció 1, print integer
syscall           # Escriu en consola $a0
```

Aquest exemple imprimirà en la consola 25 que és el valor contingut en $a0.

2.3.1.2 Activitat 2

Llegir de teclat i imprimir en consola. Per a llegir del teclat s'utilitza la **funció 5** (`read integer`). Si fiquem 5 en el registre *$v0* es llig un enter del teclat. El valor llegit es trobarà en *$v0*.

```
###################################
#                                 #
#     Codi de l'activitat 2       #
# Llegir valor introduït per teclat#
# i imprimir-lo en la consola     #
#                                 #
###################################

.text
addi $v0,$0,5     # Funció 5, read integer
syscall           # Valor llegit en $v0


addi $a0,$v0,0    # Movem el valor llegit a $a0
addi $v0,$0,1     # Funció 1, print integer
syscall           # Escriu en consola $a0
```

Aquest programa utilitza la funció 5 i llig del teclat un valor i el guarda en $v0. A continuació movem aquest valor a $a0 per a escriure'l en pantalla utilitzant la funció 1.

2.3.1.3 Activitat 3

Finalitzar el programa. Incorporem **funció 10** (`exit`). El que fa aquesta funció és eixir del procés en què es troba o acabar el programa. D'ara endavant l'utilitzarem sempre per a finalitzar els programes.

```
###################################
#                                 #
#     Codi de l'activitat 3       #
# Llegir valor introduït per teclat#
# imprimir-lo en la consola       #
# i acabar el programa            #
#                                 #
###################################

.text
addi $v0,$0,5     # Funció 5, read integer
syscall           # Valor llegit en $v0

addi $a0,$v0,0 #Movem el valor a escriure a $a0
addi $v0,$0,1     # Funció 1, print integer
syscall           # Escriu en consola $a0

addi $v0,$0,10    # Funció 10, exit
syscall            Acaba el programa
```

Qüestió 1

- Fes el codi que llig un valor *x* del teclat i escriu *x*+1 en la consola.

Qüestió 2

- Fes el codi que llig un valor *x* del teclat i escriu *x*-1 en la consola.

2.3.2 El Joc d'instruccions: Aritmètica d'enters

Les operacions del MIPS amb la Unitat Aritmètica i Lògica (ALU) utilitzen 3 operands, dos dels quals són fonts d'entrada a l'ALU i un tercer d'eixida que és el resultat de l'operació realitzada. Pel seu disseny, el MIPS sols permet que els operands de l'ALU estiguen en el processador. La destinació del resultat de l'operació serà sempre un registre, pel que fa als operands font, un dels quals ha d'estar sempre en un registre, però l'altre pot provenir d'un registre o ser un valor constant proporcionat per la instrucció. Això dona lloc a dos tipus d'instruccions

aritmètiques i lògiques depenent d'on es troben els operands font. De moment ens centrarem en les instruccions aritmètiques d'enters.

En la pràctica 1 heu estudiat la instrucció suma *addi*, en la qual un dels operands fonts i l'operand destinació són registres, i l'altre operand font és una constant: `addi rt, rs, K`.

Vegem ara l'operador suma del MIPS que utilitza 3 registres com a paràmetres, s'anomena `add` i té aquesta sintaxi:

```
add $rd, $rs, $rt
```

En la figura 1 es mostra un esquema de l'operació:

Figura 1. Representació gràfica de la instrucció add

El que fa aquesta instrucció es sumar el contingut dels registres *rs* i *rt* i emmagatzema el resultat en *rd.*

Qüestió 3

- Com s'escriu la instrucció que fa $t2=$t1+$t0?

Una altra instrucció amb l'operador suma amb tres registres és *addu*. Aquesta instrucció té un significat similar a la instrucció *addiu* ja estudiada en la pràctica 1.

Amb l'operador resta amb tres registres es troben les instruccions *sub* i *subu*. En la taula 1 podeu veure el recull de les instruccions esmentades:

Cal notar que no apareix en la taula la instrucció *subi,* ja que no forma part del conjunt d'instruccions del MIPS.

Instrucció	Exemple	Significat	Comentaris

add	add Rd,Rs,Rt	Rd ← Rs + Rt	Suma Rs i Rt i col·loca el resultat en Rd
addi	addi Rd,Rs,K	Rd ← Rs + K	Suma Rs i una constant i col·loca el resultat en Rd
addu	addu Rd,Rs,Rt	Rd ← Rs + Rt	Suma Rs i Rt i col·loca el resultat en Rd. Assumeix valors sense signe.
addiu	addiu Rd,Rs,K	Rd ← Rs + K	Suma Rs i una constant i col·loca el resultat en Rd. Assumeix valors sense signe.
sub	sub Rd,Rs,Rt	Rd ← Rs – Rt	Resta Rs menys Rt i col·loca el resultat en Rd
subu	subu Rd,Rs,Rt	Rd ← Rs – Rt	Resta Rs menys Rt i col·loca el resultat en Rd. Assumeix valors sense signe.

Taula 1. Instrucció de suma i resta del MIPS

2.3.2.1 Activitat 4

Escriviu i observeu el codi de partida següent sobre aritmètica d'enters

```
##################################
#                                #
#     Codi de l'activitat 4      #
#     Aritmètica d'enters        #
#                                #
##################################

.text 0x00400000
addiu $t0, $zero, 25
addiu $t1, $zero, 5
sub $t2,$t0,$t1

addi $v0, $zero, 10  #Eixir del programa
syscall
```

Feu una anàlisi prèvia del codi:

- Què fa cada línia del codi de partida?
- En quina adreça de memòria s'emmagatzema la instrucció sub?

Qüestió 4

- Assembla i executa el codi de l'activitat 4. Quin és el valor final del registre $t2?

2.3.3 Introducció al format d'instruccions del MIPS: codi de màquina

Ja hem comentat que l'assemblador tradueix els programes escrits en assemblador en un llenguatge intel·ligible per la màquina, que no és més que cadenes de zeros i uns. Això ho fa codificant cadascuna de les instruccions atenent el format que segueixen.

Totes les instruccions MIPS requereixen 32 bits per a codificar-se (1 paraula= 4 bytes= 32 bits). Els sis bits superiors de la paraula, es a dir, els bits de més a l'esquerra (bits 26 -31) contenen el codi d'operació que especifica quina operació farà la instrucció. Hi ha 2^6=64 codis d'operació diferents, encara que, com anirem veient al llarg de les pràctiques, hi ha més de 64 operacions distintes. La resta de bits de la paraula s'organitzen en camps per a especificar els operands de la instrucció.

Hi ha tres formats d'instrucció anomenats tipus R, I i J, encara que per ara introduirem sols els formats R i I.

Les instruccions addi i addiu segueixen un mateix format d'instrucció, el format tipus I. Aquestes instruccions utilitzen com a operands font un registre i una dada immediata i com a operand destinació un registre.

La forma general de codificació del format tipus I és la que es mostra en la figura 2.

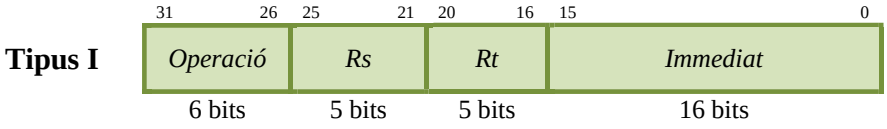

Figura 2. Codificació del format tipus I.

Com hem dit, el primer camp, es a dir els 6 bits superiors (26-31) de la paraula, és el codi d'operació i especifica quina operació realitzarà la instrucció. Els dos camps següents especifiquen els registres mitjançant 5 bits perquè disposem de 32 registres. *Rs* és el registre font i *Rt* és el registre destinació. Els restants 16 bits els utilitzem per a especificar la dada immediata.

2.3.3.1 Activitat 5

- Observeu el codi de partida *Aritmètica d'enters* de l'activitat 4. Com es codifica la primera instrucció? Feu-ho a mà (el codi d'operació de la instrucció addiu és 0x09)

- Confirmeu amb el programa assemblat que el codi màquina és el mateix.

Les instruccions que utilitzen 3 registres com a paràmetres segueixen un format d'instrucció diferent. La instrucció `add $t1, $s1, $s2` es codifica com mostra la figura 3:

Codi op (6 bits)	***Rs (5 bits)***	***Rt (5 bits)***	***Rd (5 bits)***	***Shamt (5 bits)***	***Funció (6 bits)***
A-L	$s1	$s2	$t1	-	suma
0	17	18	9	0	32
0000 00	10001	10010	0 1001	0 0000	10 0000

Figura 3. Codificació de la instrucció add $t1, $s1, $s2

En hexadecimal quedaria com: `0x02324020`.

El primer camp correspon al codi d'operació e indica que es tracta d'una operació aritmètica o lògica. Totes les instruccions aritmètiques o lògiques tenen sempre el valor 000000 per al codi d'operació. Els següents tres camps corresponen als tres registres operands, *Rs* i *Rt* són els registres fonts i *Rd* especifica al registre destí. El cinquè camp de 5 bits (shamt) no té sentit en les instruccions de suma o resta i el seu valor serà sempre 0. El camp funció indica l'operació aritmètica o lògica a realitzar (suma, resta...), en el nostre cas una suma i per tant el valor és 32 (0x20).

Qüestió 5

- Com es codifica l'última instrucció de resta del codi de partida *Aritmètica d'enters* de l'activitat 4? Feu-ho a mà i escriviu cadascun dels camps del format (el camp funció de la resta és 0x22)

- Confirmeu amb el programa assemblat que el codi màquina és el mateix.

La forma general del format tipus R és el que es mostra en la figura 4:

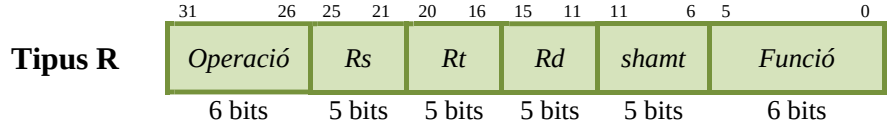

Figura 4. Codificació del format tipus R.

Com hem dit, els 6 bits superiors (26-31) de la paraula corresponen al codi d'operació, aquest camp té sempre el valor 0x0 per al format R. Els següents 3 camps especifiquen els registres fonts *Rs*, *Rt* i el registre destí *Rd* mitjançant 5 bits. El cinquè camp, *shamt*, és la quantitat de desplaçament. L'últim camp de 6 bits és la *Funció* i ens serveix per a distingir les diferents instruccions que tenen format tipus R.

Qüestió 6

- Noteu que hi ha 64 instruccions diferents amb format tipus R. Per què?

Qüestió 7

- Escriviu el codi que fa aquestes accions fent ús de les instruccions estudiades:

  $t0=5
  $t1=$t0+10
  $t2=$t0+$t1
  $t3=$t1-30

- Assembleu i executeu-lo i comproveu que el contingut dels registres és correcte.

- A la vista del codi escrit, és necessari que forme part del repertori d'instruccions la instrucció *subi*?

Qüestió 8

- Feu el codi que llig dos nombres x i y i obté per pantalla el valor de la suma *x+y*.

Qüestió 9

- Feu el codi que llig dos nombres x i y i obté per pantalla *x-y*.

2.3.4 Jocs instruccions: instruccions lògiques

El MIPS incorpora un conjunt d'operadors lògics que permeten operar amb els bits individuals dels operands fonts. Les instruccions d'operacions lògiques poden operar amb els dos operands registres fonts i en eixe cas segueixen el format tipus R o amb un operand font registre i un valor immediat com l'altre operand font. En eixe cas segueixen el format tipus I. Les instruccions lògiques són: `and`, `andi`, `or`, `ori`, `xor`, `xori` `nor`.

Fins ara heu utilitzat la instrucció addi per a ficar un valor immediat en un registre: `addi rd, $zero, immediat`.

Qüestió 10

- Podríem utilitzar la instrucció lògica *ori* per a donar un valor inicial a un registre en lloc de la instrucció *addi*?
- Com escriuríeu la instrucció que fa $t2=7 utilitzant *ori*?

Qüestió 11

- Seria o no avantatjós quant al cost temporal el fet d'utilitzar l'operador lògic *ori* en lloc de la instrucció *addi*? Raoneu-ho. (Teniu en ment els operadors aritmètics de la unitat aritmètica i lògica del processador).

Qüestió 12

- Reescriviu el codi de partida *Aritmètica d'enters* de l'activitat 4 canviant *addi* o *addiu* per *ori*:

- Quin és el valor del codi d'operació de la instrucció *ori*?

En la taula 2 es recullen les instruccions lògiques del MIPS amb format tipus R i amb format tipus I.

Instrucció	Exemple	Significat	Comentaris
and	and Rd,Rs,Rt	Rd ← Rs & Rt	3 registres operands; AND lògica
or	or Rd,Rs,Rt	Rd ← Rs | Rt	3 registres operands; OR lògica
xor	xor Rd,Rs,Rt	Rd ← Rs ⊕ Rt	3 registres operands; XOR lògica
nor	nor Rd,Rs,Rt	Rd ← (Rs | Rt)	3 registres operands; NOR lògica
and immediata	andi Rd,Rs,10	Rd ← Rs & 10	AND lògica registre i constant
or immediata	ori Rd,Rs,10	Rd ← Rs | 10	OR lògica registre i constant
xor immediata	xori Rd, Rs,10	Rd ← Rs ⊕10	XOR lògica registre constant

Taula 2. Instruccions lògiques del MIPS.

Qüestió 13

- Escriviu el codi que faça l'operació lògica OR de $t1 i $t2 i el guardeu en $t3, l'operació lògica AND de $t1 i $t2 i la guardeu en $t4 i l'operació lògica XOR de $t1 i $t2 i la guardeu en $t5. Escriviu en la finestra de registres, després d'assemblat, els següents valors per als registres $t1=0x55555555 i $t2= 0xAAAAAAAA. Executa el codi i estudia els resultats.

Qüestió 14

- Suposeu que $t1=0x0000FACE, utilitzant únicament les instruccions lògiques de la taula anterior escriviu el codi que reordene els bits de $t1 de manera que en $t2 aparega el valor 0x0000CAFE. Assembleu i escriviu en la finestra de registres $t1=0x0000FACE. Executeu i comproveu que el codi és correcte.

2.3.5 Ajudes a la programació: més sobre entrada i eixida

Fins ara heu vist que podeu llegir de teclat i escriure en la consola valors enters amb la instrucció syscall utilitzant les funcions 1 i 5. També podeu escriure per consola valors hexadecimals mitjançant la **funció 34** i valors en binari mitjançant la **funció 35** (mireu en `Mars Help> MIPS>Syscalls`).

2.3.5.1 Activitat 6

Proveu el codi següent que permet escriure en consola valors representats en hexadecimal.

```
        ###################################
        #                                 #
        #     Codi de l'activitat 6       #
        #     Imprimir en consola         #
        #      valors hexadecimals        #
        #                                 #
        ###################################

    .text 0x00400000
    ori $a0,$0,0xABC   #En $a0 el valor a escriure
    ori $v0,$0,34      #Funció 34, print hexadecimal
    syscall            #Escriu en consola el valor $a0
```

Qüestió 15

- Modifiqueu el codi de l'últim exercici de l'apartat 1.3.4 per a que aparega en la pantalla el contingut del registre $t2=0000CAFE.

Qüestió 16

- Escriviu el codi que llig un valor enter per teclat i escriviu el mateix valor en binari per la consola.

2.4 Resum

- Hi ha instruccions que fan la suma, la resta i operacions lògiques.
- Les instruccions aritmètiques i lògiques tenen tres operands però hi ha instruccions en que els tres operands són tots registres i altres en que un dels operands és un valor constant.
- Hi ha una instrucció, syscall, que permet realitzar operacions de entrada i eixida.