Prácticas AOC.- Sesión 0 Actividad.- Repaso instrucciones salto condicional y comparación (1 hora)

Nombre y apellidos: Ricardo Meza Díaz

Transferencia de datos

Sintaxis	**Tipo**	**Descripción**
clear rt	**P**	**rt = 0**
la rt,direccion	**P**	**rt = direccion**
lb rt,desp(rs)	I	rt = ext_signo(Mem[desp+rs]7..0,32)
lbu rt,desp(rs)	I	rt = ext_ceros(Mem[desp+rs]7..0,32)
ld rt,direccion	**P**	**rt = Mem[direccion]; rt+1 = Mem[direccion+4]**
lh rt,desp(rs)	I	rt = ext_signo(Mem[desp+rs]15..0,32)
lhu rt,desp(rs)	I	rt = ext_ceros(Mem[desp+rs]15..0,32)
li rd,inm32	**P**	**rd = inm32**
lui rd,inm16	I	rd31..16 = inm16; rd15..0 = 0
lw rt,desp(rs)	I	rt = Mem[desp+rs]
lwcZ rt,desp(rs)	I	coprocesadorZ(rt) = Mem[desp+rs]
lwl rt,desp(rs)	I	rt31.16 = Mem[desp+rs]
lwr rt,desp(rs)	I	rt15..0 = Mem[desp+rs]
sb rt,desp(rs)	I	Mem[desp+rs] = rt7..0
sd rt,direccion	**P**	**Mem[direccion] = rt; Mem[direccion+4] = r(t+1)**
sh rt,desp(rs)	I	Mem[desp+rs] = rt15..0
sw rt,desp(rs)	I	Mem[desp+rs] = rt
swcZ rt,desp(rs)	I	Mem[desp+rs] = coprocesadorZ(rt)
swl rt,desp(rs)	I	Mem[desp+rs] = rt31.16
swr rt,desp(rs)	I	Mem[desp+rs] = rt15..0
ulh rt,direccion	**P**	**rt = ext_signo(Mem[direccion]15..0,32)**
ulhu rt,direccion	**P**	**rt = ext_ceros(Mem[direccion]15..0,32)**
ulw rt,direccion	**P**	**rt = Mem[direccion]**
ush rt,direccion	**P**	**Mem[direccion] = rt15..0**
usw rt,direccion	**P**	**Mem[direccion] = rt**

En todas las pseudoinstrucciones, direccion puede ser una etiqueta o desp(rs).

Las instrucciones se pueden convertir en pseudoinstrucciones poniendo como segundo operando una etiqueta.

Prácticas AOC.- Sesión 0 Actividad.- Repaso instrucciones salto condicional y comparación (1 hora)

Nombre y apellidos: Ricardo Meza Díaz

Transferencia entre registros

Sintaxis	Tipo	Descripción
mfcZ rt,rd	I	rd = rt; rd: registro UCP; rt: registro del coprocesador Z
mfc1.d rd,fs	**P**	**rd = fs; r(d+1) = f(s+1); fs y f(s+1) registros de coma flotante**
mfhi rd	I	rd = hi
mflo rd	I	rd = lo
move rd,rs	**P**	**rd = rs**
mtcZ rd,rt	I	rd = rt; rd: registro UCP; rt: registro del coprocesador Z
mthi rd	I	hi = rd
mtlo rd	I	lo = rd

Aritmética para enteros

Sintaxis	Tipo	Descripción
abs rd,rs	**P**	**rd = abs(rs)**
add rd,rs,rt	I	rd = rs+rt
addi rd,rs,inm16	I	rd = rs+ext_signo(inm16,32)
addu rd,rs,rt	I	rd = rs+rt
addiu rd,rs,inm	I	rd = rs+ext_signo(inm16,32)
div rs,rt	I	lo = rs/rt; hi = rem(rs/rt)
div rd,rs1,s2	**P**	**rd = lo = rs1/s2; hi = rem(rs1,s2)**
divu rs,rt	I	lo = rs/rt; hi = rem(rs/rt)
divu rd,rs1,s2	**P**	**rd = lo = rs1/s2; hi = rem(rs1,s2)**
mul rd,rs1,s2	**P**	**hi-lo = rs1*s2; rd = lo**
mulo rd,rs1,s2	**P**	**hi-lo = rs1*s2; rd = lo**
mulou rd,rs1,s2	**P**	**hi-lo = rs1*s2; rd = lo**
mult rs,rt	I	hi-lo = rs*rt
multu rs,rt	I	hi-lo = rs*rt
neg rd,rs	**P**	**rd = -rs**
negu rd,rs	**P**	**rd = -rs**
rem rd,rs,rt	**P**	**lo = rs/rt; rd = hi = rem(rs,rt)**
remu rd,rs,rt	**P**	**lo = rs/rt; rd = hi = rem(rs,rt)**
sub rd,rs,rt	I	rd = rs-rt

Nombre y apellidos: Ricardo Meza Díaz

Instrucciones lógicas

Sintaxis	Tipo	Descripción
and rd,rs,rt	I	rd = rs AND rt
andi rd,rs,inm16	I	rd = rs AND ext_ceros(inm16,32)
nor rd,rs,rt	I	rd = rs NOR rt
not rd,rs	**P**	**rd = NOT rs**
or rd,rs,rt	I	rd = rs OR rt
ori rd,rs,inm16	I	rd = rs OR ext_ceros(inm16,32)
xor rd,rs,rt	I	rd = rs XOR rt
xori rd,rs,inm16	I	rd = rs XOR ext_ceros(inm16,32)

Instrucciones de activación condicional

Sintaxis	Tipo	Descripción
seq rd,rs1,s2	**P**	**Si rs1 = s2, rd = 1; si no, rd = 0**
sge rd,rs1,s2	**P**	**Si rs1 >= s2, rd = 1; si no, rd = 0**
sgeu rd,rs1,s2	**P**	**Si rs1 >= s2, rd = 1; si no, rd = 0**
sgt rd,rs1,s2	**P**	**Si rs1 > s2, rd = 1; si no, rd = 0**
sgtu rd,rs1,s2	**P**	**Si rs1 > s2, rd = 1; si no, rd = 0**
sle rd,rs1,s2	**P**	**Si rs1 <= s2, rd = 1; si no, rd = 0**
sleu rd,rs1,s2	**P**	**Si rs1 <= s2, rd = 1; si no, rd = 0**
slt rd,rs,rt	I	Si rs < rt, rd = 1; si no, rd = 0
slti rd,rs,inm16	I	Si rs < ext_signo(inm16,32), rd = 1; si no, rd = 0
sltu rd,rs,rt	I	Si rs < rt, rd = 1; si no, rd = 0
sltiu rd,rs,inm16	I	Si rs < ext_signo(inm16,32), rd = 1; si no, rd = 0
sne rd,rs1,s2	**P**	**Si rs1 = s2, rd = 1; si no, rd = 0**

En pseudoinstrucciones, s2 puede ser un registro o un dato inmediato de 16 bits con extensión de signo.

Instrucciones de rotación y desplazamiento

Sintaxis	Tipo	Descripción
rol rd,rt,s2	**P**	**rd = rotacion(rt,s2,izquierda)**
ror rd,rt,s2	**P**	**rd = rotacion(rt,s2,derecha)**
sll rd,rt,shamt5	I	rd = desp_log(rt,shamt5,izquierda)
sllv rd,rt,rs	I	rd = desp_log(rt,rs4..0,izquierda)
sra rd,rt,shamt5	I	rd = desp_arit(rt,shamt5,derecha)
srav rd,rt,rs	I	rd = desp_arit(rt,rs4..0,derecha)
srl rd,rt,shamt5	I	rd = desp_log(rt,shamt5,derecha)
srlv rd,rt,rs	I	rd = desp_log(rt,rs4..0,derecha)

Nombre y apellidos: Ricardo Meza Díaz

Instrucciones de control de programa

Sintaxis	Tipo	Descripción
b etiqueta	**P**	**Ramificar a etiqueta**
bcZf etiqueta	I	Si flag(coprocesadorZ) = 0, ramificar a etiqueta
bcZt etiqueta	I	Si flag(coprocesadorZ) = 1, ramificar a etiqueta
beq rs,rt,etiqueta	I	Si rs = rt, ramificar a etiqueta
beqz rd,etiqueta	**P**	**Si rd = 0, ramificar a etiqueta**
bge rs1,s2,etiqueta	**P**	**Si rs1 >= s2, ramificar a etiqueta**
bgeu rs1,s2,etiqueta	**P**	**Si rs1 >= s2, ramificar a etiqueta**
bgez rs,etiqueta	I	Si rd >= 0, ramificar a etiqueta
bgezal rs,etiqueta	I	Si rd >= 0, ramificar a etiqueta y enlazar ($ra = PC)
bgt rs1,s2,etiqueta	**P**	**Si rs1 >= s2, ramificar a etiqueta**
bgtu rs1,s2,etiqueta	**P**	**Si rs1 >= s2, ramificar a etiqueta**
bgtz rs,etiqueta	I	Si rd > 0, ramificar a etiqueta
ble rs1,s2,etiqueta	**P**	**Si rs1 <= s2, ramificar a etiqueta**
bleu rs1,s2,etiqueta	**P**	**Si rs1 <= s2, ramificar a etiqueta**
blez rs,etiqueta	I	Si rd <= 0, ramificar a etiqueta
blt rs1,s2,etiqueta	**P**	**Si rs1 < s2, ramificar a etiqueta**
bltu rs1,s2,etiqueta	**P**	**Si rs1 < s2, ramificar a etiqueta**
bltz rs,etiqueta	I	Si rd < 0, ramificar a etiqueta
bltzal rs,etiqueta	I	Si rd < 0, ramificar a etiqueta y enlazar ($ra = PC)
bne rs,rt,etiqueta	I	Si rs <> rt, ramificar a etiqueta
bnez rd,etiqueta	**P**	**Si rd <> 0, ramificar a etiqueta**
j objetivo	I	PC = PC31..28 || (objetivo << 2)
jal objetivo	I	ra = PC; PC = PC31..28 || (objetivo << 2)
jalr rs,rd	I	rd = PC; PC = rs
jr rs	I	PC = rs

Misceláneas

Sintaxis	Tipo	Descripción
rfe	I	Restaurar desde excepción (Restaura el registro Status)
syscall	I	Llamada a un servicio del sistema ($v0: número del servicio)
break codigo20	I	Provoca una excepción (código 1 reservado para el depurador)
nop	I	No operación

Prácticas AOC.- Sesión 0 Actividad.- Repaso instrucciones salto condicional y comparación (1 hora)

Nombre y apellidos: Ricardo Meza Díaz

1.- Para cada pseudoinstrucción de la siguiente tabla, escriba una secuencia mínima de instrucciones MIPS que realicen la misma tarea (BIG es un número que requiere ser representado con 32 bits y SMALL un número que puede representarse con 16 bits).

Pseudoinstrucción	Tarea	Solución
move $t1, $t2	**$t1 = $t2**	add $t1, $t1, $t2 (suponemos que $t1 está inicializado a 0)
clear $t0	**$t0 = 0**	sub $t0, $t0, $t0
beq $t1, SMALL, L	**if ($t1 == SMALL) ir a L**	sub $t1, $t1, SMALL bcZt L #Si flag(coprocesadorZ) = 1, ramificar a etiqueta
beq $t2, BIG, L	**if ($t2 == BIG) ir a L**	sub $t2, $t2, BIG bcZt L
li $t1, SMALL	**$t1 = SMALL**	add $t1, $t1, SMALL (Suponemos que $t1 está inicializada a 0)
li $t2, BIG	**$t2 = BIG**	add $t2, $t2, BIG (Suponemos que $t2 está inicializada a 0)
ble $t3, $t5, L **#Salto menor o igual**	**if ($t3 <= $t5) ir a L**	sub $t5, $t5, $t3 bgez $t5, L
bgt $t4, $t5, L **#Salto a mayor**	**if ($t4 > $t5) ir a L**	sub $t4, $t4, $t5 bgtz $t4, L
bge $t5, $t3, L	**if ($t5 >= $t3) ir a L**	sub $t5, $t5, $t3

#Salto mayor o igual		bgez $t5, L
addi $t0, $t2, BIG	**$t0 = $t2 + BIG**	add $t0, $t2, BIG
lw $t5, BIG($t2)	**$t5 = MEM[$t2 + BIG]**	add $t2, $t2, BIG lw $t5, $t2

Nombre y apellidos: Ricardo Meza Díaz

2.- Escriba una única sentencia de C que implemente el siguiente código MIPS. Asuma que $1 almacena A, $2 almacena B y $3 almacena C.

```
        beq $1, $0, False
        beq $2, $0, False
        addi $3, $0, 1
        j Continue
False:  addi $3, $0, 0
Continue: …
```

Solución:

beq $1, $0, Continue
beq $2, $0, Continue
addi $3, $0, 1
Continue: …

3.- Escriba un programa en ensamblador MIPS que implemente la siguiente sentencia C. Almacene A en $1, B en $2, C en $3 y D en $4.

D = C || (A && B);

Solución:

```
.data
A:   .word 5
B:   .word 3
C:   .word 4
D:   .word 7

#-------------------------------------
    .text
    .globl main
main:
    lw $s1, A
    lw $s2, B
    lw $s3, C
    lw $s4, D

    sub $s4, $s4, $s4   #Inicializo $s4 a 0

    and $s1, $s1, $s2
    or $s3, $s3, $s2
```

```
add $s4, $s4, $s4

li $v0, 10          #Finalizar el programa
syscall
```

4.- Escriba un programa en ensamblador MIPS que implemente la siguiente sentencia C. Almacene A en $1, B en $2, C en $3 y D en $4.

D = C || (!A && B);

Solución:

```
.data
A:      .word 5
B:      .word 3
C:      .word 4
D:      .word 7

#-------------------------------------
        .text
        .globl main
main:
        lw $s1, A
        lw $s2, B
        lw $s3, C
        lw $s4, D

        sub $s4, $s4, $s4   #Inicializo $s4 a 0

        not $s1, $s1
        and $s1, $s1, $s2
        or $s3, $s3, $s1
        add $s4, $s4, $s4

        li $v0, 10                  #Finalizar el programa
        syscall
```

Debe entregar un fichero con extensión .s o .asm con la solución de cada ejercicio (3 y 4). Debe además adjuntar el código en este fichero de texto.