Arquitectura de Computadoras: *Una Definición*

- ✓ *Computer architecture is the abstraction of a physical system (microcode and hardware) as seen by the machine-language programmer or a compiler writer. It is the definition of the conceptual structure and fuctional behavior of a processor as opposed to such attributes as the processor's underlying data flow and controls, logic design, and circuit technology."*

From: Myers, G.J., "Advances in Computer Architecture," Second Edition, John Wiley & Sons, New York, N.Y., 1978.

1

Aspectos Arquitecturales

- ✓ Instrucciones
- ✓ Modos de acceso a operandos
- ✓ Formatos de instrucciones
- ✓ Tipos de datos
- ✓ Memoria Primaria
- ✓ Registros
- ✓ Interrupciones
- ✓ Flujo de ejecución de instrucciones
- ✓ Stacks

2

Aspectos no Arquitecturales

- ✓ Memoria secondaria
- ✓ Buses
- ✓ Periferales
- ✓ Unidad de control
- ✓ ALU
- ✓ Tecnología de fabricación
- ✓ Circuitos lógicos

3

Aspectos que Podrían ser Arquitecturales

- ✓ Caches
- ✓ Pipelines
- ✓ I/O

4

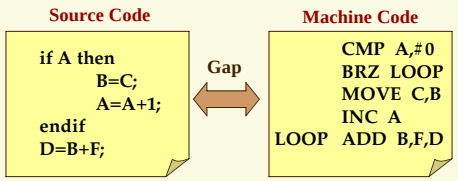
Reduced Instruction Set Computers (RISC)

Principio:

Apoyar aquellas estructuras de lenguajes de alto nivel que son utilizadas con frecuencia y que tienen un impacto considerable en el tiempo de ejecución de programas. El apoyo a estas estructuras debe basarse en elementos arquitecturales primitivos que faciliten la implementación de un procesador rápido, eficiente y sencillo.

5

Semantic Gap



6

Microprocesadore CISC Contemporáneos

- ✓ **Pentium**

Microprocesadores RISC

- ✓ **Pioneros**
  - *IBM 801*
  - *Berkeley RISC I*
  - *Stanford MIPS*
- ✓ **Contemporáneos**
  - *SPARC (SUN)*
  - *MIPS*
  - *HP Precision*
  - *Alpha*
  - *PowerPC*
  - *IBM R6000*

COMPLEX INSTRUCTION SET COMPUTERS (CISC)

Principio:

Proveer apoyo a diferentes lenguajes de alto nivel moviendo funciones complejas al "hardaware". Facilitar el desraorll oedc moipaloder secrrnaode ls"menait cag"p .L saa qriuettcrusaC Sl Cons gieu nnu aifolosíf anep raitucal.r lEn morb eedC Sl Cesl sea jdduci óapard siitgniurlas de las arquitecturas RISC.

Características RISC

- ✓ **Primordiales**
  - *Un número grande de registros*
  - *Operaciones se llevan a cabo con operandos mantenidos en registros*
  - *Los accesos a memoria se realizan exclusivamentemediante instruciones load/store*
- ✓ **Otros**
  - *Instruciones de tamaño fijo*
  - *Pocos formatos de instrucciones*
  - *Pocos y sencillos modos de acceso a operandos*

Falacias sobre Arquitecturas RISC

- ✓ **Tienen un número reducido de instrucciones**
- ✓ **Las instrucciones son sencillas**

Características No Exclusivas de Arquitecturas RISC

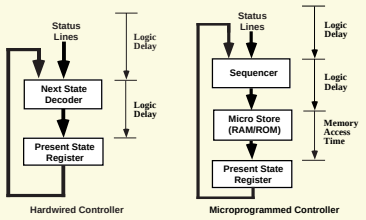
- ✓ **Delayed branches**
- ✓ **Register Windows**
- ✓ **Una instrucción por ciclo**

Implicaciones de Arquitecturas RISC sobre la Iplementación

- ✓ Ciclos cortos
- ✓ Más apropiadas para implementación de unidad de control “hard wired”
- ✓ Más apropiadas para implementación de “pipelines”
- ✓ Dependen más de compiladores

13

Unidades de Control “Hardwired” vs. Microprogramada



14

SPARC: *Registros*

- *PC - Program Counter*
- *nPC - Next Program Counter*
- *IR - Instruction Register*
- *Y - Multiply Step Register*
- *TBR - Trap Base Register*
- *WIM - Window Invalid Mask*
- *CWP - Current Window Pointer*
- *PSR - Processor Status Register*
- *r0-r31 - Integer Registers*
- *f0-f31 - Floating Point Registers*

15

SPARC: *Registros de Números Enteros*

r0-r7 (g0-g7) - registros globales

r8-r15 (o0-o7) - registros de salida

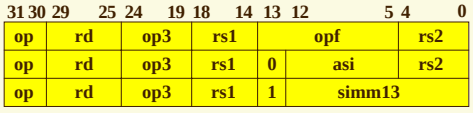
r16-r23 (l0-l7) - registros locales

r24-r31 (i0-i7) - registros de entrada

16

SPARC: *Formatos de Instrucciones*

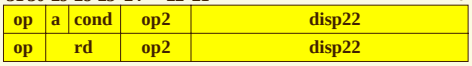
- ✓ Floating Point
- ✓ Data Movement
- ✓ ALU



17

SPARC: *Formatos de Instrucciones*

- ✓ branches
- ✓ sethi



- ✓ call

18

SPARC: *Claves para Formatos de Instrucciones*

- op - opcode formato 1
- op2 - opcode formato 2
- op2 - opcode formato 3
- rs1 - registro fuente 1
- rs2 - registro fuente 2
- rd - registro destino
- simm13 - número inmediato con signo
- i - registro o valor inmediato
- opf - sub-opcode para punto flotante
- disp22 - desplazamiento de 22 bits
- disp30 - desplazamiento de 30 bits
- a - annul bit para branches
- cond - selección de condición para branches

SPARC: *Memoria Primaria*

- 2^{32} *bytes*
- *Tamaños:*
  - Bytes (8 bits)
  - Halfword (16 bits)
  - Word (32 bits)
  - Doubleword (64 bits)
- *Big-endian*

Address	Data
300	D0
301	F0
302	C7
303	0A

La localización 300 tiene el valor de D0F0C70A

SPARC: *Instrucciones load/store*

Mnemonic	op3	Meaning
ldsb	001001	Load signed byte
ldsh	001010	Load signed halfword
ld	001000	Load word
ldub	000001	Load unsigned bytes
lduh	000010	Load unsigned halfword
ldd	000011	Load doubleword
stb	000101	Store byte
sth	000110	Store halfword
st	000100	Store word
std	000111	Store doubleword
swap	001111	Swap register with memory word

SPARC: *Instrucciones load/store*

- *op = 11*
- *Localización efectiva:*
  - [rs1] + [rs2] /el contenido de rs1 + el contenido de rs2
  - [rs1] + simm13 /el contenido de rs1 + valor inmediato simm13
- *Ejemplos de "assembler":*

Instrucción	rs1	rs2	rd	simm13
ld [%r4 + %r5], %r6	4	5	6	
ld [%r4 - 12], %r1	4		1	-12
st %r2, [%r3 + 300]	3		2	300
st %r1, [%r6 + %r5]	6	5	1	

SPARC: *Ejemplos de Instrucciones load/store*

Instrucción		r1	r4	r5	loc18	loc62	loc304
ld [%r4 + %r5], %r1	antes	40	12	50	360	15	2000
	después	15	12	50	360	15	2000
ld [%r4 - 12], %r5	antes	40	30	63	360	15	2000
	después	40	30	360	360	15	2000
st %r1, [%r5 + 300]	antes	40	30	4	360	15	2000
	después	40	30	4	360	15	40

SPARC: *Intrucciones Aritméticas y Lógicas*

Mnemonic	op3	Meaning
add	0S0000	add
addx	0S1000	add with carry
sub	0S0100	substract
subx	0S1100	substract with carry
and	0S0001	and bitwise
andn	0S0101	nand bitwise
or	0S0010	or bitwise
orn	0S0110	nor bitwise
xor	0S0011	xor bitwise
xorn	0S0111	xnor bitwise
sll	100101	shift left logical by rs2 or simm13
srl	100110	shift right logical by rs2 or simm13
sra	100111	shift right arithmetic by rs2 or simm13

SPARC: *Intrucciones Aritméticas y Lógicas*

- *op = 10*
- *S - set condition code bit (S=1 permite alterar CC)*
- *Ejemplos de "assembler":*

Instrucción	rs1	rs2	rd	simm13
add %r7, %r5, %r6	7	5	6	
subcc %r4, - 30, %r1	4		1	-30
sub %r2, %r3, %r7	2	3	7	
sll %r1, %r6, %r5	1	6	5	

Nota: Para permitir alterar los "condition_____
_____
do S=1)

25

SPARC: *Ejemplos de Instrucciones Ariméticas y Lógicas*

Instrucción		r1	r4	r5
add %r4, %r5, %r1	antes	40	12	50
	después	62	12	50
sub %r4, - 12, %r5	antes	40	30	63
	después	40	30	42
sra %r1, %r5, %r4	antes	- 128	4	4
	después	-128	- 8	4

26

SPARC: *Instrucciones de Brinco Condicional*

Mnemonic	op	op2	cond	Meaning
ba	00	010	1000	Brinca siempre
be	00	010	0001	Brinca si igual
bne	00	010	1001	Brinca si no igual
ble	00	010	0010	Brinca si menor o igual
bcc	00	010	1101	Brinca si carry=0
bcs	00	010	0101	Brinca si carry=1
bneg	00	010	0110	Brinca si negativo
bvc	00	010	1111	Brinca si overflow=0
bvs	00	010	0111	Brinca si overflow=1

27

SPARC: *Instrucciones de Brinco Condicional*

- *Si a=1 (el annul bit) y se da el brinco la instrucción que sigue el branch se ejecuta .*
- *Si a=1 y no se da el brinco la instrucción que sigue el branch se anula (no se ejecuta) .*
- *Si a=0 la instrucción que sigue el branch siempre se ejecuta .*
- *La localización efectiva del brinco es PC + 4*disp22*
- *Ejemplos de "assembler":*

Instrucción	disp22
be +300	+300
bneg, a, -120	-120 (a=1)

28

SPARC: *Ejemplos de Instrucciones de Brinco Condicional*

Loc	Instrucción	Secuencia del PC brinca	Secuencia del PC no brinca
1000	be +300	1000	1000
		1004	1004
		1300	1008
4000	bneg, a, -120	4000	4000
		4004	4008
		3880	4012

29

SPARC: *Instruccions de Subrutinas*

Mnemonic	op	op3	Meaning
call	01		Brinca y guarda PC en r15
jmpl	10	111000	Brinca y guarda PC en rd

30

SPARC: *Instruccions de Subrutinas*

- *Las instrucciones de subrutinas son de tipo "delayed"*
- *La localización efectiva de call es PC + 4*disp30*
- *La localización efectiva de jmpl es rs1 + rs2 o rs1 + simm13*

Ejemplos de "assembler"

```
call    4008
jmpl    %r4   %r6    %r1
jmpl    %r4    12    %r0
```

31

SPARC: *Ejemplos de Instrucciones de Subrutinas*

Instrucción		r0	r1	r4	r6	r15	PC	nPC
call 4008	antes	0	12	50	48	50	300	304
	después	0	12	50	48	304	304	4008
jmpl %r4, %r6, %r1	antes	0	12	50	48	50	300	304
	después	0	304	50	48	50	304	98
jmpl %r4, 12, %r0	antes	0	12	50	48	50	300	304
	después	0	12	50	48	50	304	62

32

SPARC: *Instrucciones de Manejo de Ventanas del Register Windows*

Mnemonic	op	op3	Meaning
save	10	111100	Provee nueva ventana y suma CWP = CWP-1
restore	10	111101	Restablece vieja ventana y suma CWP = CWP+1

Ejemplos de "assembler":

```
save %r4, %r5, %r1
restore %r4, -30, %r1
```

Ambas instrucciones activan una nueva ventana de registros y en adición realizan una operación de suma que envuelve los operandos rs1, rs2, rd y simm13.

33

SPARC: *Ejemplos de Instrucciones de Manejo de Ventanas del Register Windows*

Instrucción		r1	r4	r5	CWP
save %r4, %r5, %r1	antes	40	12	50	5
	después	62	12	50	4
restore %r4, - 12, %r5	antes	40	30	63	4
	después	40	30	18	5

34

SPARC: *Otras Instrucciones*

Mnemonic	op	op2	Meaning
sethi	00	100	Carga valor inmediato de 22 bits en los 22 bits más significativos de rd. Los restantes 10 bit menos significativos se toman el valor de 0.

Ejemplo:

Instrucción		r1
sethi 3FFFFFh, %r1	antes	ACC3D5B0h
	después	FFFFFC00h

35

SPARC: *Sintetización de Instrucciones*

nop ➡ **sethi 0, %r0**

ret ➡ **jmpl %r5, %r21, %r0**

move ➡ **or %r0, %r1, %r6**

36

SPARC: *Sintetización de Modos de Acceso a Operandos*

register	add %r5, %r21, %r0
inmediate	add %r5, +34, %r3
base-indexed	ld %r5, %r21, %r2
displacement	ld %r5, -15, %r6
register indirect	st %r5, %r0, %r6
absolute	st %r0, 1004, %r6
relative	ba +3004