Estructura de Computadores

Grado de Ingeniería Informática ETSINF

Tema 1: El procesador

Objetivos

 Contextualizar la asignatura

✓ Conocer el contexto de la arquitectura MIPS32

✓ Conocer los aspectos más generales de la arquitectura MIPS32

✓ Aprender el ciclo de ejecución del procesador  Conocer el diseño de la ruta de datos basada en

multiplexores  Conocer el diseño de la unidad de control del procesador

2

Contenido y Bibliografía

 Introducción  1 – Arquitectura MIPS32

Introductorio ✓ Características básicas

(Tema 0)

✓ Ejemplo de ejecución  2 – La ruta de datos y la unidad de control

✓ Etapas de búsqueda y decodificación

✓ Diseño de la ruta para aritmético/lógicas de tipo R

✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I

✓ Diseño de la ruta para instrucciones lw/sw

✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4) 3

Introducción

Procesador Memoria

Bus de interconexión

 Componentes básicos de un computador

✓ Procesador (UCP)

Entrada/Salida

✓ Memoria

✓ Entrada/Salida

✓ Interconexión

4

Concepto de programa almacenado:

- Las instrucciones se representan como números - Los programas son almacenados en memoria para ser leídos o escritos también como números - Se interactúa con el mundo exterior mediante dispositivos de Entrada/Salida

Introducción: Tipos de computadores

**Tipo Potencia**

**Ancho Palabra (bits) Consumo Aplicaciones**

Gestión en Corporaciones, **Supercomputadores** Muy alta potencia

de cálculo. 64, 128 Extremadamente alto

Gobiernos. Aplicaciones computación masiva: centros de investigación.

**Estaciones de trabajo** Alta 32, 64 Alto Diseño, simulación, control:

Empresas, Universidades

**Portátiles** Alta-Media 32, 64 Medio

Bajo

5 Informática en general Computadores personales

**Tabletas** Media 32,64 Muy bajo Informática personal

**Sistemas empotrados** Media-Alta. 16, 32, 64 Medio

Bajo

Sistemas empotrados para: Comunicaciones (routers, switches). Electrodomésticos, automoción, periféricos, sistemas industriales,...

**Teléfonos** Media-Baja 16, 32 Extremadamente bajo Teléfonos móviles (Smartphones)

Introducción: El mercado de los procesadores

**Ventas 2015**

MPUs Tabletas 6%

MPU: Microprocessor Unit

Todo procesador se define de acuerdo con una **ARQUITECTURA**

6

MPUs telefonía mobil 26%

PCs - servidores - supercomput. 56%

MPUs sistemas empotrados

11% Otras CPUs

1%

Introducción: Arquitectura

 Arquitectura (ISA: Instruction Set Architecture)

 Hace referencia al repertorio de instrucciones, registros, modelo de excepciones,

manejo de la memoria virtual, mapa de direcciones físicas y otras características

comunes de un procesador

 Todo aquello que el programador debe saber acerca de la máquina  Ejemplos de arquitecturas:

**Arquitectura Empresas**

x86, x86-64, IA-32, INTEL®64 INTEL, AMD,..

MIPS-32, MIPS-64 Mips Technologies

DEC Alpha Architecture Digital Equipment Corp.

Power , PowerPC Apple, IBM, Motorola

ARM Architecture Advanced RISC Machines Holdings)

 Las arquitecturas pueden evolucionar, dando lugar a versiones de 16, 32 o 64

bits, siempre compatibles entre sí.

7

Introducción: Implementación

 Implementación: Hace referencia a las características concretas de los

circuitos que conforman el diseño de un procesador concreto que ejecuta una

arquitectura.

 También se denominan **Microarquitecturas**

**Arquitectura Microarquitecuras Modelos Fabricantes**

MIPS MIPS32 ;MIPS64,

Warrior, Aptiv

8 R2000, R3000, R4000,... P5600, P6600, MicroAptiv, InterAptiv, ProAptiv MIPS Tech.

Imagination Tech.

IA-32, Intel 64 P5, P6 Pentium II, III, Pro INTEL

NetBurst Pentium 4, Pentium D, Celeron D

Pentium M Core Duo, Core Solo, Pentium M, Celeron M

Intel Core Core 2 Duo, Quad, Extreme, Xeon 5xxx, 7xxx,

Celeron dual-core

Atom Atom

Core i i3, i5, i7

X86, x86-64 K5,K6,K8 Athlon 64, Phenom, Turion, Semprom, Opteron AMD

ALPHA EV4,EV5,..EV9 21x64, 22x64, 23x64 (x = 0,1,2,3,4) DEC

Arquitectura MIPS

 MIPS: **M**icroprocerssor without **I**nterlocked **P**ipeline **S**tages)

✓ Procesador RISC (**R**educed **I**ntruction **S**et) desarrollado en la Universidad de

Stanford por John L. Hennessy que emplea la técnica de segmentación .

✓ Comercializado por MIPS Technologies (Silicon Graphics International)

✓ Arquitectura sencilla → Muy utilizada por las universidades para docencia

✓ El diseño segmentado del MIPS es el precursor de la mayoría de los

procesadores RISC posteriores

**Versión juego de instrucciones (ISA) Ancho palabra Procesadores**

MIPS I 32 R2000, R3000

MIPS II 32 R6000

MIPS III 64 R4000

MIPS IV 64 R5000, R10000

MIPS V 64 -

MIPS32 32 4K

MIPS64 64 5K

9

Implementaciones del MIPS

 Los diferentes modelos del MIPS, R2000, R3000,... R10000 fueron utilizados como CPUs de

computadores estándar por SGI, Olivetti, Siemens, etc.. No tuvo demasiado éxito, en un mercado

dominado por Intel y AMD, excepto por el procesador Loongson (China)

 En la década de los 90 MIPS Technologies licenció el diseño del MIPS en forma de dos arquitecturas:

MIPS32 y MIPS64. Estas fueron ampliamente usadas en microcomputadores y sistemas empotrados

diseñados y fabricados por terceras marcas.

 En 2012, ImaginationTech. adquirió MIPS Tech. Desde entonces los diseños MIPS se han distribuido

ampliamente en forma de IP-cores (Warrior, Aptiv), siendo un ‘peso-pesado’ en el mercado de los

procesadores para sistemas empotrados y SoC: Cisco, Sony, NEC, Microchip, Toshiba,....

IP-core (Itellectual Property core): Bloque lógico o circuito que es propiedad intelectual de una compañía pero que puede ser licenciado a otras compañías que lo pueden usar en FPGAs o diseños ASIC (Applicacion Specific Integrated Circuit) Embedded System: Sistema computador embebido o empotrado en una máquina y diseñado para realizar unas funciones específicas, usualmente de tiempo real. SoC (System On Chip): Circuito integrado que integra todos los componentes de un computador o sistema electrónico en un solo chip.

10

Implementaciones del MIPS

 Los diferentes modelos (cores) del MIPS se pueden ver en:

 https://en.wikipedia.org/wiki/List\_of\_MIPS\_microarchitectures

 https://imgtec.com/mips/architectures

 El mercado de los IP-cores en 2013 se distribuía como indica

la figura:

11

**IP-cores market share 2013**

ARM (Cortex cores)

28,20%

Synopsys Inc. (ARC cores) 43,20% 5,10%

Imagination Tech. (MIPS cores)

9%

13,20% Cadence Design Systems (8051

cores) Otros

Arquitectura MIPS32: Características Básicas

 Máquina RISC (Reduced Instruction Set Computer)  Ancho de palabra y tamaño de buses de 32 bits  Principales tamaño de datos en las instrucciones:

✓ Byte (B), halfword (H), word (W)

 Arquitectura de carga/almacenamiento

✓ Instrucciones específicas de lectura (carga) y escritura (almacenamiento) en

memoria

✓ Todos los operandos de una instrucción aritmética se cargan inicialmente

en registros, o están en la propia instrucción (constantes)

✓ Instrucciones aritméticas con 3 operandos de 32 bits en registros  Modos de funcionamiento: usuario, núcleo (kernel) y depuración

12

Arquitectura MIPS32: Características Básicas

CPU

Coprocesador 1 (FPU)

registros

registros $0

.$0 .. .

$31$31LO HI

Coprocesador 0

Procesador Memoria

registros

BadVAddr

Cause

Bus de interconexión Status

EPC

El registro PC (Program Counter) no pertenece a la arquitectura pero se utilizan en la CPU. PC apunta a la instrucción a buscar en memoria para su

Entrada/Salida

ejecución. El PC no es visible al programador, pero es modificado por ciertas instrucciones de forma indirecta.

13

Contenido memoria

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

La arquitectura MIPS32 será utilizada como ejemplo en toda la asignatura, utilizando siempre ejemplos de código en ensamblador del MIPS

Arquitectura MIPS32 en la asignatura

CPU

Coprocesador 1 (FPU)

registros

registros $0

.$0 .. .

$31$31Tema 3

LO HI

Coprocesador 0

Procesador Memoria

registros

BadVAddr

Cause

Bus de interconexión Status

EPC

Entrada/Salida

14

Tema 4

Temas 1&2

Tema 11

Temas 7-10

Temas 5&6

Contenido memoria

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

Arquitectura MIPS32: Modelo de programación

 CPU

✓ 32 registros de 32 bits ($0..$31)

✓ Registros HI-LO

 Coprocesador 0

✓ Control del sistema (jerarquía, excepciones, modos de ejecución, ...)

✓ 4 registros específicos (hay más)

 FPU (Coprocesador 1, opcional)

✓ 32 registros de 32 bits ($0..$31)

✓ Aritmética coma flotante de simple precisión (32 registros)

✓ Aritmética coma flotante de doble precisión (16 parejas de regs)

15

Arquitectura MIPS32: Registros

**Nombre Registro Uso**

CPU

Coprocesador 1 (FPU)

$zero $0 Constante 0

registros

registros $at $1 Registro temporal del ensamblador

$0

.$0 .$v0-$v1 $2-$3 Retorno de funciones

.

.

$a0-$a3 $4-$7 Argumentos de funciones $31$31$t0-$t7 $8-$15 Registros temporales

$s0-$s7 $16-$23 Temporales salvados

$t8-$t9 $24-$25 Registros temporales

LO HI

$k0-$k1 $26-$27 Utilizados por el SO

$gp $28 Puntero global

Coprocesador 0

$sp $29 Puntero de pila

registros

$fp $30 Puntero de marco (frame)

BadVAddr

Cause

$ra $31 Dirección de retorno

Status

EPC

$f0-$f31 $0..$31 Registros de coma flotante

16

Arquitectura MIPS32: Memoria

0x00000000

0x80000000

0xFFFFFFFF

17

Espacio de usuario (2GB)

Código

0x10000000

Espacio de sistema (2GB)

Espacio de direccionamiento

Datos

Pila

0x00400000

0x80000000

Arquitectura MIPS32: Memoria

 Directivas de ensamblador

✓ Directivas de segmentos de memoria

 .data [dirección]

 .text [dirección]

 .end

✓ Directivas de reserva de memoria de datos

 .space n

 .byte b1 [, b2] ...

 .half b1 [, b2] ...

 .word b1 [, b2] ...

 .ascii cadena1 [, cadena2] ...

 .asciiz cadena1 [, cadena2]...

18

Las variables del programa son codificadas como etiquetas de datos A = 0x10000000 W = 0x10000004 ...

.data 0x1000000 A: .byte 2, 3, 4 W: .word 33 V: .space 100

.data 0x10004000 C1: .ascii “hola” C2: .asciiz “hola”

.text 0x00400000 .....end

Arquitectura MIPS32: Memoria

 Direccionamiento

✓ Direccionamiento a nivel de byte

✓ Modos big-endian y little-endian soportados  Alineamiento

✓ Byte en cualquier dirección

✓ Media palabra (half) en dirección par

✓ Palabra (word) en dirección múltiplo de 4

✓ Doble palabra (dword) en dirección múltiplo de 8  Directiva

✓ .align N

✓ Alinea a partir de la siguiente instrucción múltiplo de 2N

19

Arquitectura MIPS32: Instrucciones

 Grupos de instrucciones

✓ Transferencia entre registros

✓ Carga/almacenamiento

Las instrucciones serán ejercitadas en las sesiones de laboratorio. El apéndice contiene ejemplos de uso.

✓ Aritméticas

✓ Lógicas

✓ Comparación

✓ Salto

✓ Coma flotante

✓ Otras  Sintaxis y codificación

**Instrucciones vs pseudo-instrucciones** Las instrucciones son soportadas por el hardware mientras que las pseudo-instrucciones son traducidas por el ensamblador en un conjunto de instrucciones equivalente EJEMPLO: la $t1, 0x10002000 -> lui $1, 0x1000

ori $t1, $1, 0x2000

✓ http://www.mrc.uidaho.edu/mrc/people/jff/digital/MIPSir.html

✓ http://en.wikipedia.org/wiki/MIPS\_architecture

20

Arquitectura MIPS32: ejemplo de ejecución

.data .data 0x10002000 0x10002000 a: a: .byte .byte 0x80, 0x80, 0xFF, 0xFF, 0xA0, 0xA0, 0x00 0x00 b: b: .word .word -1 -1

.text .text 0x00400000 0x00400000 lui lui $1, $1, 4096 4096 ori ori $8, $8, $1, $1, 8192 8192 lb lb $9, $9, 0($8) 0($8) lb lb $10, $10, 2($8) 2($8) add add $11, $11, $9, $9, $10 $10 sb sb $11, $11, 3($8) 3($8) add add $9, $9, $0, $0, $0 $0

Pseudo instrucción

Ensamblado:

•Pseudoinstrucciones

•Nombres de registros

•Cálculos de valores

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

Pseudo

.text 0x00400000 instrucción

la $t0, a lb $t1, 0($t0) lb $t2, 2($t0)

Alias Alias en en nombre nombre de de los los registros registros add $t3, $t1, $t2 sb $t3, 3($t0) add $t1, $0, $0

.end

Contenido memoria

Carga

*Obtenido con PCSpim*

[00400000] 3c011000 lui $1, 4096 [a] ; 6: la $t0, a [00400004] 34282000 ori $8, $1, 8192 [a] [00400008] 81090000 lb $9, 0($8) ; 7: lb $t1, 0($t0) [0040000c] 810a0002 lb $10, 2($8) ; 8: lb $t2, 2($t0) [00400010] 012a5820 add $11, $9, $10 ; 9: add $t3, $t1, $t2 [00400014] a10b0003 sb $11, 3($8) ; 10: sb $t3, 3($t0) [00400018] 00004820 add $9,$0,$0 ; 189: add $t1, $0,$0

*Codif. Formato R*

0011 1100 0000 0001 0001 0000 0000 0000

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

cop Rs Rt Rd func

**31 25 20 15 10**

**5 0**

0000 0001 0010 1010 0101 1000 0010 0000

Instrucción con formato R (add $11, $9, $10) C.op: bits 31-26: 000000 Rs: bits 25-21: 01001 -> $9 Rt: bits 20-16: 01010 -> $10 Rd: bits 15:11: 01011 -> $11 Func: bits 5-0: 100000 -> add

21

PC = 0x00400000 RI = 0x3c011000.data 0x10002000

a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000

**la $t0, a** lb $t1, 0($t0) lb $t2, 2($t0) add $t3, $t1, $t2 sb $t3, 3($t0) add $t1, $0, $0

PC = 0x00400000

$1 = 0x10000000

Copr CPU

1 (FPU)

registros

$0 ..

$31LO HI

BadVAddr Status

registros $0 ..

$31ALU

ALU

PC PC

RI

Coprocesador 0

registros

Cause EPC

Ejemplo: Ejecución instrucción 1

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

22

Dirección Contenido .data 0x10002000

0x00400000 0x3c011000 a: .byte 0x80, 0xFF, 0xA0, 0x00

0x00400004 0x34282000 c: .word -1

0x00400008 0x81090000 .text 0x00400000

0x0040000C 0x810a0002 **lui $1, 4096**

0x00400010 0x012a5820 ori $8, $1,8192

0x00400014 0xa10b0003 lb $9, 0($8)

0x00400018 0x00004820 lb $10, 2($8) add $11, $9, $10 sb $11, 3($8)

...

add $9, $0, $0

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

La instrucción leída se ubica en el registro de instrucción RI

lui $1, 4096

Lectura de la instrucción ubicada en 0x00400000

Contenido memoria

PC PC = = 0x00400004 0x00400004 RI = 0x34282000

$1 = 0x10000000

$1 = 0x10000000 $8 = 0x10002000

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 **la $t0, a** lb $t1, 0($t0) lb $t2, 2($t0) add $t3, $t1, $t2 sb $t3, 3($t0) add $t1, $0, $0

Copr CPU

1 (FPU)

registros

registros $0

.$0 .. .

$31$31ALU ALU LO HI PC PC

RI

Coprocesador 0

registros BadVAddr Status

Cause EPC

Ejemplo: Ejecución instrucción 2

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

23

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 lui $1, 4096 **ori $8, $1,8192** lb $9, 0($8) lb $10, 2($8) add $11, $9, $10 sb $11, 3($8) add $9, $0, $0

ori $8, $1, 8192

Lectura de la instrucción ubicada en 0x00400004

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

Contenido memoria

PC PC = = 0x00400008 0x00400008 RI = 0x81090000

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 la $t0, a **lb $t1, 0($t0)** lb $t2, 2($t0) add $t3, $t1, $t2 sb $t3, 3($t0) add $t1, $0, $0 .end

$1 $1 = = 0x10000000 0x10000000 $8 $8 = = 0x10002000 0x10002000 $9 = 0xFFFFFF80

Lectura del byte con direccíón $t0+0 y almacenamiento en $t1

Copr CPU

1 (FPU)

registros

registros $0

.$0 .. .

$31$31ALU ALU LO HI PC PC

RI

Coprocesador 0

registros BadVAddr Status

Cause EPC

Ejemplo: Ejecución instrucción 3

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

24

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 lui $1, 4096 ori $8, $1,8192 **lb $9, 0($8)** lb $10, 2($8) add $11, $9, $10 sb $11, 3($8) add $t1, $0, $0

Lectura Lectura del del byte byte ubicado en en 0x10002000 que es 0x80, y se extenderá el signo g al almacenarlo ( (0xFFFFFF80) )

lb $9, 0($8)

Lectura de la instrucción ubicada en 0x00400008

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

Contenido memoria

PC = 0x0040000C RI = 0x810a0002

$1 = 0x10000000 $8 = 0x10002000 $9 = 0xffffff80 $10= 0xffffffa0

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 lui $1, 4096 ori $8, $1,8192 lb $9, 0($8) **lb $10, 2($8)** add $11, $9, $10 sb $11, 3($8) add $9, $0, $0

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 la $t0, a lb $t1, 0($t0) **lb $t2, 2($t0)** add $t3, $t1, $t2 sb $t3, 3($t0) add $t1, $0, $0

Copr CPU

1 (FPU)

registros

registros $0

.$0 .. .

$31$31ALU ALU LO HI PC PC

RI

Coprocesador 0

registros BadVAddr Status

Cause EPC

Ejemplo: Ejecución instrucción 4

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

25

Lectura del byte ubicado en 0x10002002, que es 0xa0, y se extiende el signo al almacenarlo (0xFFFFFFA0)

lb $10, 2($8)

Lectura de la instrucción ubicada en 0x0040000C

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

Contenido memoria

PC = 0x00400010 RI = 0x012a5820

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 la $t0, a lb $t1, 0($t0) lb $t2, 2($t0) **add $t3, $t1, $t2** sb $t3, 3($t0) add $t1, $0, $0

$1 = 0x10000000 $8 = 0x10002000 $9 = 0xffffff80 $10= 0xffffffa0 $11= 0xffffff20

Copr CPU

1 (FPU)

registros

registros $0

.$0 .. .

$31$31ALU ALU LO HI PC PC

RI

Coprocesador 0

registros BadVAddr Status

Cause EPC

Ejemplo: Ejecución instrucción 5

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

26

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 lui $1, 4096 ori $8, $1,8192 lb $9, 0($8) lb $10, 2($8) **add $11, $9, $10** sb $11, 3($8) add $9, $0, $0

add $11, $9, $10

Lectura de la instrucción ubicada en 0x00400010

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x00a0ff80 0x10002004 0xffffffff ...

Contenido memoria

PC = 0x00400014 RI = 0xa10b0003

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 la $t0, a lb $t1, 0($t0) lb $t2, 2($t0) add $t3, $t1, $t2 **sb $t3, 3($t0)** add $t1, $0, $0

$1 = 0x10000000 $8 = 0x10002000 $9 = 0xffffff80 $10= 0xffffffa0 $11= 0xffffff20

Almacena el byte de menor peso de $t3 en memoria, en la dirección resultado de sumar $t0 y 3

Copr CPU

1 (FPU)

registros

registros $0

.$0 .. .

$31$31ALU ALU LO HI PC PC

RI

Coprocesador 0

registros BadVAddr Status

Cause EPC

Ejemplo: Ejecución instrucción 6

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

27

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 lui $1, 4096 ori $8, $1,8192 lb $9, 0($8) lb $10, 2($8) add $11, $9, $10 **sb $11, 3($8)** add $9, $0, $0

Escritura de byte en la dirección 0x10002003, Con el byte de menos peso de $11, que es 0x20

sb $11, 3($8)

Lectura de la instrucción ubicada en 0x00400014

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x20a0ff80 0x10002004 0xffffffff

...

Contenido memoria

PC = 0x00400018 RI = 0x00004820

$1 = 0x10000000 $8 = 0x10002000 $9 = 0x00000000 $10= 0xffffffa0 $11= 0xffffff20

OperaciónADD con [$0] y[$0]

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 la $t0, a lb $t1, 0($t0) lb $t2, 2($t0) add $t3, $t1, $t2 sb $t3, 3($t0) **add $t1, $0, $0**

Copr CPU

1 (FPU)

registros

registros $0 $0

.$0 .. .

$31 $31 $31ALU ALU LO HI PC PC

RI

Coprocesador 0

registros BadVAddr Status

Cause EPC

Ejemplo: Ejecución instrucción 7

RI

Procesador Memoria

Bus de interconexión

Entrada/Salida

28

.data 0x10002000 a: .byte 0x80, 0xFF, 0xA0, 0x00 c: .word -1

.text 0x00400000 lui $1, 4096 ori $8, $1,8192 lb $9, 0($8) lb $10, 2($8) add $11, $9, $10

sb $11, 3($8) **add $9, $0, $0**

add $9, $0, $0

Lectura de la instrucción ubicada en 0x00400018

Dirección Contenido 0x00400000 0x3c011000 0x00400004 0x34282000 0x00400008 0x81090000 0x0040000C 0x810a0002 0x00400010 0x012a5820 0x00400014 0xa10b0003 0x00400018 0x00004820 ...

0x10002000 0x20a0ff80 0x10002004 0xffffffff ...

Contenido memoria

Contenido y Bibliografía

 Introducción  1 – Arquitectura MIPS32

✓ Características básicas

✓ Ejemplo de ejecución  2 – La ruta de datos y la unidad de control

✓ Etapas de búsqueda y decodificación

✓ Diseño de la ruta para aritmético/lógicas de tipo R

✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I

✓ Diseño de la ruta para instrucciones lw/sw

✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4) 29

La ruta de los datos y la unidad de control

 Diseño del procesador

 Elección del juego de instrucciones a ejecutar

✓ Subconjunto del juego de instrucciones del MIPS

 Aritméticas y comparación: add, sub, and, or, slt

 Carga/almacenamiento: lw, sw

 Salto: beq, jump (la instrucción jump será añadida posteriormente)

 Ciclo único de reloj

30

La ruta de los datos y la unidad de control

 Diseño del procesador

✓ Combinación de elementos lógicos sencillos

 Circuitos combinacionales: puertas, decodificadores, multiplexores, ALU, ...

 Circuitos secuenciales: biestables, registros, banco de registros, memoria, ...

✓ Las **prestaciones** dependen del número de instrucciones que ejecuta

por unidad de tiempo y de la duración del ciclo de reloj

 **Objetivo inicial: Ejecutar una instrucción por ciclo y reducir el ciclo**

**de reloj**

31

La ruta de los datos y la unidad de control

 Etapas en el diseño

✓ Diseñar la ruta de datos

 Seleccionar los elementos necesarios para la ejecución de las instrucciones

 Interconectar los elementos

✓ Diseñar la unidad de control

 Identificar las señales de control necesarias

 Diseñar la lógica asociada a cada señal

32

Contenido y Bibliografía

 Introducción  1 – Arquitectura MIPS32

✓ Características básicas

✓ Ejemplo de ejecución  2 – La ruta de datos y la unidad de control

✓ Etapas de búsqueda y decodificación

✓ Diseño de la ruta para aritmético/lógicas de tipo R

✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I

✓ Diseño de la ruta para instrucciones lw/sw

✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4) 33

Etapas de búsqueda y decodificación

Todas las instrucciones comienzan con su búsqueda en la

memoria. La dirección se encuentra en el PC.

RI ←Mem[PC] Una vez la instrucción está en RI la unidad de control la

decodifica para conocer cuál es y ejecutarla.

 Elementos funcionales necesarios

 Ruta de los datos

 Funcionamiento

34

Etapas de búsqueda y decodificación

OPERACIONES:  Leer instrucción empleando PC y almacenar en RI

 Incrementar PC en 4

 Decodificar la instrucción en RI

ELEMENTOS:  Registro PC y RI y memoria de instrucciones.

 Unidad sumadora.

 Campos COP y Función del RI a la unidad de control

35

Memoria registro PC El RI se transparente, diseña como

es

RI decir actualiza su contenido

en cuanto 32

cambia su entrada. 32 La señal de escritura Dir estará

Dato permanentemente activa.

El contenido de los registros PC y RI es accesible sin necesidad de señal de

LInst

lectura.

36

+4

PC

32

32

PCin

La señal PCin es la patilla de escritura del PC

✓ Será activa por flanco

37

Instrucción Aritmética tipo R

add rd, rs, rt

RI

COP RS RT RD Función

Unidad de control Indica que es una operación ARITMETICA

¿Cuál de ellas?

add sub and orslt

38

Instrucciones tipo I

addi rt, rs, inmediato

RI

COP RS RT inmediato

Identifica la instrucción

Unidad de control

39

+4

Memoria

PC RI

32

32 Dir Dato 32

PCin Cop y Fun.

LInst

Unidad de control

40

+4

PC Memoria

RI

32

Dir Dato

LInst

La unidad de control

Tiempo para ejecutar la instruc ción 32 32

Flanco y Actualizo PC

PCIn Cop y Fun. Reloj Add $3, $2,$1

41

+4

PC Memoria

RI

32 32

32 Dir Dato

Flanco y

LInst Actualizo PC

Reloj

La unidad de control PCIn

Cop y Fun.

42

Contenido y Bibliografía

 Introducción  1 – Arquitectura MIPS32

✓ Características básicas

✓ Ejemplo de ejecución  2 – La ruta de datos y la unidad de control

✓ Etapas de búsqueda y decodificación

✓ Diseño de la ruta para aritmético/lógicas de tipo R

✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I

✓ Diseño de la ruta para instrucciones lw/sw

✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4) 43

La ruta de los datos aritmético/lógicas tipo R

add/sub/and/or/slt $3, $1, $2

COP RS RT RD FUNC. 000000 00001 00010 00011

000000

tipo

 ¿Cómo se ejecutan estas instrucciones?

RD = RS operado RT

 Elementos funcionales necesarios

 Ruta de los datos

 Funcionamiento de la ruta

 Señales de control

44

Arquitectura MIPS32. Op. aritmética tipo R

CPU registros $00add $3, $1, $2 $13$210$3 13 0

..

Lectura registros $1 y $2

$31

Suma 3 010

Escritura resultado en $3

13

HI LO

45

Add rd, rs, rt

OPERACIONES:  Leer rs, rt

 Sumar rs y rt

 Guardar en rt dato sumado

ELEMENTOS:  Banco de registros con dos direcciones de lectura  UAL

 Banco de registros, con una tercera dirección de escritura

46

RI

Banco Registros

5

Dir1 Dato1

32

Dir2

Dato2

LReg

5

32

47

RSRT

3232

U.A.L.

Unidad aritmético y lógica

32 OpALU3La UAL tendrá que tener las cinco operaciones: suma, resta, and, or y slt

La entrada OpALU tendrá tres bits, para poder codificar en binario las cinco operaciones

**OpALU Operación** 000 a ∧ b (and) 001 a ∨ b (or) 010 a + b (suma aritmética) 110 a – b (resta) *111 a < b (menor que)*

48

RI

Banco Registros

532 Dir1 Dato1 5Dir2 5

32Dato2

Dir3 32

Dato3

EReg

LReg

Resultado del Cálculo

49

RI

Banco Registros

532 Dir1 Dato1

La señal **LReg**, por nivel: la lectura delos registros 5Dir2 5

32Dato2

durará todo el ciclo

Dir3 32

Resultado Dato3

del Cálculo

EReg

LReg

La señal **EReg**, por flanco: la escritura se realiza al final del ciclo, permitiendo así leer y escribir en el mismo ciclo en el mismo registro: ej. add $3, $3, $2

50

Ruta de los datos aritmético/lógicas tipo R 32

32

3232 PCIn

Dato3

Reloj

RI +4

PC

Memoria

Banco Registros

5Dir1

Dir Dato

5Dir2

Dato1

5

Dato2

LInst

LReg

323232

A.L.U.

Dir3

OpALUDato3

EReg

3

Cop y Fun.

La unidad de control

51

Se actualiza el PC

Reloj

+4

Memoria

RI

PC

Banco Registros

32 32

Dir Dato

32

5Dir1 $3 = 15

Dato1

32

5Dir2

5

Dir3

Dato2

32U.A.L.

Op

PCIn

Dato3 32LInst

Flanco y Actualizo PC

EReg

LReg 3

010 (+)

Cop y Fun.

La unidad de control

Add $3, $2,$1

52

¿Cuál sería el T.Ciclo mínimo?

 Escritura en registros PC y RI

sin coste.

 Lectura en memoria, 20ns. 20 Lectura banco de registros, 1515ns. 5 Unidad de control 5ns en

decodificar.

 UAL 5ns todas las operaciones.

 Escritura en registro, 20ns las

520

señales estables a la entrada del banco de registros.

6553 Todas las instrucciones del MIPS leen en el banco de registros, sin saber qué instrucción se está ejecutando, se puede acceder al mismo y ahorrar tiempo

20Max(15,5)

520

60

+4

Memoria

RI

Banco Registros

PC

32

32

32 5Dir1 Dato1 Dir Dato

5

Dir2

32

3 LInst

LReg

Cop y Fun.

32

RELOJ

¿ Tabla de verdad ?

Dato2

U.A.L.

Dir3

¿Cómo

PCIn

5

genera los flancos?

Dato3 32 EReg

La unidad de control

32

Op

54

Aritmético/lógicas Tipo R: Señales de control

**Mem. Instr.**

55 **Banco Registros ALU CP**

Instrucción Cop Función PCIn LInst LReg EReg OpALU

add rd, rs, rt 000000 100000 1 1 1 1 010

sub rd, rs, rt 000000 100010 1 1 1 1 110

and rd, rs, rt 000000 100100 1 1 1 1 000

or rd, rs, rt 000000 100101 1 1 1 1 001

slt rd, rs, rt) 000000 101010 1 1 1 1 111

Entradas Salidas

**OpALU Operación** 000 a ∧ b (and) 001 a ∨ b (or) 010 a + b (suma aritmética) 110 a – b (resta) *111 a < b (menor que)*

PCIn, LMem y LReg se pueden mantener siempre activas

Contenido y Bibliografía

 Introducción  1 – Arquitectura MIPS32

✓ Características básicas

✓ Ejemplo de ejecución  2 – La ruta de datos y la unidad de control

✓ Etapas de búsqueda y decodificación

✓ Diseño de la ruta para aritmético/lógicas de tipo R

✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I

✓ Diseño de la ruta para instrucciones lw/sw

✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4) 56

La ruta de los datos aritméticas tipo I

addi/slti $3, $1, 8

COP RS RT Inmediato 16 tipo

00001

00011 0000000000001000

 ¿Cómo se ejecutan estas instrucciones?

RT = RS operado Inmediato extendido a 32b

 Elementos funcionales necesarios

 Ruta de los datos

 Funcionamiento de la ruta

 Señales de control

57

Arquitectura MIPS32. Op. Carga tipo I

CPU

3 0x10000000 0

RI

58 registros

addi $3, $2, 0x1000 $00$1$2$3

0x10001000 ..

$31

0Escritura resultado en $3

Extensor

ALU

Lectura registro $2 y dato

0x1000

0x00001000

Suma

0x10000000

0x10001000

Addi rt, rs, inmediato 16

OPERACIONES:  Leer rs, inmediato16

 Extender inmediato16 a 32b

 Realizar la operación

 Guardar en rt dato sumado

ELEMENTOS:  Banco de registros (el mismo ruta tipo R)  Unidad extensora del bit de signo.

 Multiplexor en una entrada de la UAL.  Multiplexores en entradas del Banco de registros.

59

Operación: extender desplazamiento de 16 a 32 bits RI

Extiende el bit de signo, bit 15, a los 16 bits de más peso

16

32 Unidad extensora

60

Cálculo RI

Banco Registros en UAL

5

32Dir1 Dato1

Dir2 Dato2

LReg

32 32

16

Unidad extensora

32

32

A.L.U.

5

MxALU

3

OpALU

MxALU= 0 Tipo R MxALU = 1 Tipo I

61

RI Banco Registros

5Dir1 Dato1 5

Dir2 Dato2

LReg

MxDst = 1 Tipo R MxDst = 0 Tipo I

32

325

Dir3 32

Dato3

MxDst

EReg

62

OpALU Ruta de los datos aritmético/lógicas tipo R +

32

RI

addi/slti +4

Memoria PCBanco Registros

Dir Dato

Dato2

Dato3

LReg

3

La unidad de control32

5

Dir1 Dato1

3232LInst

PCIn OpUAL

Extensor

32

32 Dir2 55

MxALU

32

MxDst

.........

MxDst

MxALU

325

PCIn

5

Dir3

Dato3

EReg

32 16

Campo Inmediato

Reloj

Tipo I: MxDst= 0 MxUAL=1 Tipo R: MxDst = 1 MxUAL=0

63

Add $3,$2,$1 # 0x00811820

32

RI

+4 Actualizar

Memoria

PC32

Dir Dato 32

32Banco Registros

5

Dir1 $3 Dato1

= 33

32

Dato2

32 Dir2 55 LInst

Dato3

MxALU

32

MxDst

LReg

16

Extensor Campo Inmediato

325

5

Dir3

Dato3

32

OpALU

EReg

64

Addi $3,$2,1 # 0x20430001

32

RI

+4 Actualizar

Memoria

PC32

Dir Dato 32

32Banco Registros

5

Dir1 $3 Dato1

= 11

32

Dato2

32 Dir2 55 LInst

Dato3

MxALU

32

MxDst

LReg

16

Extensor Campo Inmediato

325

5

Dir3

Dato3

32

OpALU

EReg

65

Aritméticas Tipo I: Señales de control

**Mem. Instr.**

66 **Banco Registros ALU Multiplexore CP**

Instrucción Cop Función PCin LMem LReg EReg OpALU MxALU MxDst

add rd, rs, rt 000000 100000 1 1 1 1 010 0 1

addi rt, rs, imdto16 001000 ------------- 1 1 1 1 010 1 0

slti rt, rs, inmdto16 001010 ------------- 1 1 1 1 111 1 0

Entradas Salidas

Distinto **OpALU Operación**

camino instrucciones 000 001 a ∧ b (and)

a ∨ b (or)

tipo R

010 110 a a e + – instrucciones b b (suma (resta)

aritmética)

tipo I

*111 a < b (menor que)*

Contenido y Bibliografía

 Introducción  1 – Arquitectura MIPS32

✓ Características básicas

✓ Ejemplo de ejecución  2 – La ruta de datos y la unidad de control

✓ Etapas de búsqueda y decodificación

✓ Diseño de la ruta para aritmético/lógicas de tipo R

✓ Diseño de la ruta para aritmético/lógicas R y aritméticas tipo I

✓ Diseño de la ruta para instrucciones lw/sw

✓ Diseño de la ruta para instrucciones de salto beq/bne

Bibliografía: Patterson, D.A., Hennessy, J.L., “Estructura y diseño de computadores. La interfaz hardware-Software,” 4a edición, Ed. Reverté, 2011, Cap 4 (4.1 – 4.4) 67

lw rt, rs, inmdto16 # lw $3, 0x100($2)

COP RS RT desplazamiento 100011 rs rt Desplazamiento 16

 ¿Cómo se ejecuta esta instrucción?

Rt ←Mem[Rs + desplazamiento16 extendido]

 Elementos funcionales necesarios

 Ruta de los datos

 Señales de control

68

Instrucción lw: Señales de control

**Reg. CP**

69 **Mem. Instr.**

**Banco Registros**

**ALU Memori**

**a DATOS**

**Multiplexores**

Instrucción Form Código

Op.

Función PCIn LInst LReg EReg OpALU LMem MxALU MxDst MxER

add rd, rs, rt R 000000 100000 1 1 1 1 010 0 0 1 0

addi rt, rs, inmdto16

I 001000 1 1 1 1 010 0 1 0 0

lw rt, desp(rs) I 100011 1 1 1 1 010 1 1 0 1

Entradas Salidas

**OpALU Operación** 000 a ∧ b (and) 001 a ∨ b (or) 010 a + b (suma aritmética) 110 a – b (resta) *111 a < b (menor que)*

Arquitectura MIPS32. Op. Carga tipo I

CPU

lw $3, 0x100($2)

3 0x10000000 0

RI

70 registros $00$1$2Lectura registro $2 y dato $3

47 ..

Suma y obtención dirección

$31 0Acceso a Memoria Extensor 0x0100

0x10000000

Escritura resultado en $3 0x00000100

ALU

0x10000100

47

Bus de interconexión

Memoria

OPERACIONES:  Leer rs, desplaz16

 Extender inmediato16 a 32b

 Calcular dirección

 Acceder a memoria

 Guardar en rt dato accedido

ELEMENTOS:  Banco de registros

(el mismo ruta tipo R)  Unidad extensora del bit de signo

(la misma ruta tipo I)  UAL empleando camino ruta tipo I  Memoria datos  Multiplexores en entrada del Banco de registros.

Añadir a la ruta lw rt, desplaz16(rs)

71

Acceso a la memoria DATOS

Memoria DATOS

32

32

A.L.U.

Dir Dato

OpALULMem

3

72

RTRD

Escritura en Rt del Dato de memoria

Banco Registros

Dir1 Dato1

5

Dato2 Dir2 55

5

Dir3

32

Dato3

MxDst

LReg

MxER

OpALU

Dato3

EReg

Tipo I (addi): MxDst= 0 MxER = 0 Tipo I (lw): MxDst= 0 MxER = 1 Tipo R: MxDst = 1 MxER = 0

Memoria

Dir Dato

32

LMem

32

73

Ruta de los datos arit./lóg. R, addi/slti y lw

32

Memoria Instrucciones PC

32

32PCIn

OpUAL

RI

+4

Memoria Datos

Dato2 Dir Dato

Dir Dato

LInst

LMem 32 LBReg

MxER

Banco Registros

5

Dir1 Dato1

32

32 Dir2 55

32

MxDst

MxALU

3232

La unidad de control

32

5

5

MxALU

MxDst

.........

Dir3

Dato3

OpALU

3

Dato3

32

EBReg

16

Extensor

MxER

Campo Desplazamiento

**MxDst MxER MxALU**

**lw** 0 1 1 **Tipo R** 1 0 0 **addi/slti** 0 0 1

74

sw rt, rs, inmdto16 # sw $3, 0x100($2)

COP RS RT desplazamiento 011011 rs rt Desplazamiento 16

 ¿Cómo se ejecuta esta instrucción?

Rt → Mem[Rs + desplazamiento16 extendido]

 Elementos funcionales necesarios

 Ruta de los datos

 Señales de control

75

Arquitectura MIPS32. Op. Almacenamiento

CPU

sw $3, 0x100($2)

3 0x10000000 47

RI

76 registros $00$1$2$3 ..

$31 0Extensor

ALU

Lectura registro $2,$3 y dato

0x0100

0x00000100

Suma y obtención dirección

0x10000100

Bus de interconexión

Acceso a Memoria para escribir

0x10000000

Un 47 en dir 0x10000100

47

Memoria

OPERACIONES:  Leer rs, rt y desplaz16

 Extender inmediato16 a 32b

 Calcular dirección

 Escribir en memoria el registro rt

ELEMENTOS:  Banco de registros

(el mismo ruta tipo R)  Unidad extensora del bit de signo

(la misma ruta tipo I)  UAL empleando camino ruta tipo I

 Memoria datos para escribir registro rt

Añadir a la ruta sw rt, desplaz16(rs)

77

Acceso a la memoria DATOS

Memoria DATOS

32

32 Salida ALU

Dir DatoO

32 RT

DatoI EMem

LMem

Leer: Activar LMem y al cabo de un tiempo sale dato indicado en Dir por DatoO Escribir: Activar EMem y al cabo de un tiempo se escribe el dato entrante por DatoI en la dirección indicada en Dir.

78

Ruta de los datos aritm/lóg R, addi/slti, y lw/sw

32

Memoria Instrucciones PC

32

32PCIn

OpUAL

RI

+4

Memoria Datos

Dato2 Dir Dato

Dir

LInst

LMem

LBReg

32

MxER

Banco Registros

5

Dir1 Dato1

32

32 Dir2 55

32

MxDst

MxALU

32DatoI

32

5

5

Dir3

MxALU

OpALU

MxDst

EBReg

Dato3

Dato3

32

EMem

32 16

Extensor

MxER

.........

3

Campo La unidad

Desplazamiento

**MxDst MxER MxALU**

de control

**lw** 0 1 1 **sw** X X 1 **Tipo R** 1 0 0 **addi/slti** 0 0 1

79

Instrucción sw: Señales de control

**Reg. CP**

80 **Mem. Instr.**

**Banco Registros**

**ALU Memoria**

**DATOS**

**Multiplexores**

Instrucción Form Código

Op.

Función PCin LInst LReg EReg OpALU LMem EMem MxALU MxDst MxER

add rd, rs, rt R 000000 100000 1 1 1 1 010 0 0 0 1 0

addi rt, rs, inmdto16

I 001000 1 1 1 1 010 0 0 1 0 0

lw rt, desp(rs) I 100011 1 1 1 1 010 1 0 1 0 1

sw rt, desp(rs) I 101011 1 1 1 0 010 0 1 1 x x

Entradas Salidas

**OpALU Operación** 000 a ∧ b (and) 001 a ∨ b (or) 010 a + b (suma aritmética) 110 a – b (resta) *111 a < b (menor que)*