IPL escola superior de tecnologia e gestão instituto politécnico de leiria

Sistemas Computacionais 2018/2019
Guia de consulta para os testes
Nuno Veiga, Rui Vasco

Taxa de Transferência de Dados (TTD) e Largura de Banda (LB)

LB = Largura de Dados x TTD =
= Largura de Dados x Frequência x N.º Transf. por ciclo relógio (b/s)

Evolução do *socket* para as últimas 4 microarquiteturas Intel:

Processadores Extreme	Restantes Gamas de Processadores
Socket 1366	*Socket* 1156
Socket 2011	*Socket* 1155
Socket 2011-v3	*Socket* 1150
Socket 2066	*Socket* 1151

Tabela ASCII

[ENTER]	Espaço
13	32
dH	20H

0	…	9
48	…	57
30H	…	39H

A	…	Z
65	…	90
41H	…	5aH

a	…	z
97	…	122
61H	…	7aH

IEEE-754 (Precisão simples)

Sinal	Expoente	Mantissa
1 bit	8 bits	23 bits

Lei de Amdahl

$$T_{novo} = T_{anterior} * (Fração_{não\ melhorada} + \frac{Fração_{melhorada}}{Speedup_{fração}})$$

$$Speedup_{global} = \frac{1}{Fração_{não\ melhorada} + \frac{Fração_{melhorada}}{Speedup_{fração}}}$$

Assembler directives MIPS64

.data - start of data segment
.text - start of code segment
.code - start of code segment (same as .text)
.org <n> - start address
.space <n> - leave n empty bytes
.asciiz <s> - enters zero terminated ascii string
.ascii <s> - enter ascii string
.align <n> - align to n-byte boundary
.word <n1>,<n2>.. - enters word(s) of data (64-bits)
.byte <n1>,<n2>.. - enter bytes
.word32 <n1>,<n2>.. - enters 32 bit number(s)
.word16 <n1>,<n2>.. - enters 16 bit number(s)
.double <n1>,<n2>.. - enters floating-point number(s)

Memory Mapped I/O area WinMIPS64

CONTROL: .word32 0x10000
DATA: .word32 0x10008

Set CONTROL = 1, Set DATA to Unsigned Integer to be output
Set CONTROL = 2, Set DATA to Signed Integer to be output
Set CONTROL = 3, Set DATA to Floating Point to be output
Set CONTROL = 4, Set DATA to address of string to be output
Set CONTROL = 5, RGB colour output
Set CONTROL = 6, Clears the terminal screen
Set CONTROL = 7, Clears the graphics screen
Set CONTROL = 8, read the DATA from the keyboard
(either an integer or a floating-point)
Set CONTROL = 9, read one byte from DATA, no character echo.

Instruções MIPS64

add.d freg,freg,freg	- add floating-point
and reg,reg,reg	- logical and
andi reg,reg,imm	- logical and immediate
bc1f imm	- branch to address if FP flag is FALSE
bc1t imm	- branch to address if FP flag is TRUE
beq reg,reg,imm	- branch if pair of registers are equal
beqz reg,imm	- branch if register is equal to zero
bne reg,reg,imm	- branch if pair of registers are not equal
bnez reg,imm	- branch if register is not equal to zero
c.eq.d freg,freg	- set FP flag if equal to
c.le.d freg,freg	- set FP flag if less than or equal to
c.lt.d freg,freg	- set FP flag if less than
cvt.d.l freg,freg	- convert 64-bit integer to a double FP format
cvt.l.d freg,freg	- convert double FP to a 64-bit integer format
dadd reg,reg,reg	- add integers
daddi reg,reg,imm	- add immediate
daddu reg,reg,reg	- add integers unsigned
daddui reg,reg,imm	- add immediate unsigned
ddiv reg,reg,reg	- signed integer division
ddivu reg,reg,reg	- unsigned integer division
div.d freg,freg,freg	- divide floating-point
dmul reg,reg,reg	- signed integer multiplication
dmulu reg,reg,reg	- unsigned integer multiplication
dsll reg,reg,imm	- shift left logical
dsllv reg,reg,reg	- shift left logical by variable amount
dsra reg,reg,imm	- shift right arithmetic
dsrav reg,reg,reg	- shift right arithmetic by variable amount
dsrl reg,reg,imm	- shift right logical
dsrlv reg,reg,reg	- shift right logical by variable amount
dsub reg,reg,reg	- subtract integers
dsubu reg,reg,reg	- subtract integers unsigned
halt	- stops the program
j imm	- jump to address
jal imm	- jump and link to address (call subroutine)
jalr reg	- jump and link to address in register
jr reg	- jump to address in register
l.d freg,imm(reg)	- load 64-bit floating-point
lb reg,imm(reg)	- load byte
lbu reg,imm(reg)	- load byte unsigned
ld reg,imm(reg)	- load 64-bit double-word
lh reg,imm(reg)	- load 16-bit half-word
lhu reg,imm(reg)	- load 16-bit half word unsigned
lui reg,imm	- load upper half of register immediate
lw reg,imm(reg)	- load 32-bit word
lwu reg,imm(reg)	- load 32-bit word unsigned
mfc1 reg,freg	- move data from FP register to integer register
mov.d freg,freg	- move floating-point
movn reg,reg,reg	- move if register not equal to zero
movz reg,reg,reg	- move if register equals zero
mtc1 reg,freg	- move data from integer register to FP register
mul.d freg,freg,freg	- multiply floating-point
nop	- no operation
or reg,reg,reg	- logical or
ori reg,reg,imm	- logical or immediate
s.d freg,imm(reg)	- store 64-bit floating-point
sb reg,imm(reg)	- store byte
sd reg,imm(reg)	- store 64-bit double-word
sh reg,imm(reg)	- store 16-bit half-word
slt reg,reg,reg	- set if less than
slti reg,reg,imm	- set if less than immediate
sltiu reg,reg,imm	- set if less than immediate unsigned
sltu reg,reg,reg	- set if less than unsigned
sub.d freg,freg,freg	- subtract floating-point
sw reg,imm(reg)	- store 32-bit word
xor reg,reg,reg	- logical xor
xori reg,reg,imm	- exclusive or immediate