COMPITINO #2 di *CALCOLATORI ELETTRONICI 1* del 04-12-2006 **MATRICOLA**_____

COGNOME_____

NOME_____

1) [20/40] Si supponga di dover programmare la porta seriale 16550A in modo da generare una trama di 5 bit dati, 1.5 bit di stop, parità dispari (numero di "uni" dispari) a un baud rate pari a 9600. Tale chip sia mappato a partire dall' indirizzo 0x 8000 03F8 dello spazio di memoria di un processore MIPS. Si scriva il codice assembly necessario per programmare tale chip supponendo che la frequenza esterna di clock sia pari a 1.8432 MHz.

2) [20/40] Si disegni lo schema architetturale/logico (comprensivo di multiplexer, decoder, porte logiche secondo quanto necessario) di una cache di dimensione pari a 512B, 4 vie con dimensione del blocco pari a 64B e supponendo che il bus dati di uscita sia di larghezza pari a 4B (32 bit) mentre l'indirizzo e' specificato su 4B (32 bit). La politica di rimpiazzamento e' LRU, la politica di scrittura e' writeback: si rappresenti anche l'hardware necessario ad implementare queste politiche.

MIPS instructions

Instruction	Example	Meaning	Comments
add	`add $1,$2,$3`	$1 = $2 + $3	3 operands; exception possible
subtract	`sub $1,$2,$3`	$1 = $2 - $3	3 operands; exception possible
add immediate	`addi $1,$2,100`	$1 = $2 + 100	+ constant; exception possible
subtract immediate	`subi $1,$2,100`	$1 = $2 - 100	- constant; exception possible
multiplication	`mult $1, $2`	Hi,Lo= $1 x $2	64-bit Signed Product ; result in Hi,Lo
division	`div $1, $2`	Hi= $1 % $2, Lo = $1 / $2	Signed division
move from Hi	`mfhi $1`	$1 = Hi	Create copy of Hi
move from Lo	`mflo $1`	$1 = Lo	Create copy of Lo
and	`and $1,$2,$3`	$1 = $2 & $3	3 register operands; Logical AND
or	`or $1,$2,$3`	$1 = $2 | $3	3 register operands; Logical OR
nor	`nor $1,$2,$3`	$1 = !($2 | $3)	3 register operands; Logical NOR
xor	`xor $1,$2,$3`	$1 = $2 ^ $3	3 register operands; Logical XOR
and immediate	`andi $1,$2,100`	$1 = $2 & 100	Logical AND register, constant
or immediate	`ori $1,$2,100`	$1 = $2 | 100	Logical OR register, constant
xor immediate	`xori $1,$2,100`	$1 = $2 ^ 100	Logical XOR register, constant
shift left logical	`sll $1,$2,10`	$1 = $2 << 10	Shift left by constant
shift right logical	`srl $1,$2,10`	$1 = $2 >> 10	Shift right by constant
load word	`lw $1,100($2)`	$1 = Memory[$2+100]	Data from memory to register
load byte	`lb $1,100($2)`	$1 = Memory[$2+100]	Data from memory to register
load byte unsigned	`lbu $1,100($2)`	$1 = Memory[$2+100]	Data from mem. to reg.; no sign extension
store word	`sw $1,100($2)`	Memory[$2+100] = $1	Data from register to memory
store byte	`sb $1,100($2)`	Memory[$2+100] = $1	Data from register to memory
load address	`la $1,var`	$1 = &var	Load variable address
branch on equal	`beq $1,$2,100`	if ($1 = = $2) go to PC+4+100	Equal test; PC relative branch
branch on not equal	`bne $1,$2,100`	if ($1 != $2) go to PC+4+100	Not equal test; PC relative
set on less than	`slt $1,$2,$3`	if ($2 < $3) $1 = 1; else $1 = 0	Compare less than; 2`s complement
set on less than immediate	`slti $1,$2,100`	if ($2 < 100) $1 = 1; else $1 = 0	Compare < constant; 2`s complement
set on less than unsigned	`sltu $1,$2,$3`	if ($2 < $3) $1 = 1; else $1 = 0	Compare less than; natural number
set on less than imm. unsigned	`sltiu $1,$2,100`	if ($2 < 100) $1 = 1; else $1 = 0	Compare constant; natural number
jump	`j 10000`	go to 10000	Jump to target address
jump register	`jr $31`	go to $31	For switch, procedure return
jump and link	`jal 10000`	$31 = PC + 4;go to 10000	For procedure call

UART Intel-16550A

- **Visione logica del chip (tutti registri a 8 bit):**

A2-A0	access	DLAB bit	mnemonic		
000	R	0	RBR		Receive Buffer Register Dove si trova il dato ricevuto
000	W	0	THR		Transmitter Holding Register Dove si mette il dato da trasmettere
001	RW	0	IER		Interrupt Enable Register
010	R	0	IIR		Interrupt Identification Register
010	W	0	FCR		FIFO Control Register
011	RW	-	LCR		Line Control Register
100	RW	-	MCR		Modem Control Register I bit 0 e 1 sono dei latch su /RTS e /DTS (in scrittura gli altri bit possono essere messi a 0)
101	RW	-	LSR		Line Status Register
110	RW	-	MSR		Modem Status Register
111	RW	-	SCR		Scratchpad Register Registro di appoggio da usare liberamente
001	RW	1	DLM		Divisor Latch MSB
000	RW	1	DLL		Divisor Latch LSB

Visione logica: LCR, Line Control Register