COMPITO di *CALCOLATORI ELETTRONICI 1* del 28-06-11 **MATRICOLA**_____

COGNOME_____

NOME_____

1) [34/40] Trovare il codice assembly MIPS o x86 corrispondente del seguente programma (**utilizzando solo e unicamente istruzioni dalla tabella sottostante**), **rispettando le convenzioni di utilizzazione dei registri dell'assembly MIPS** (riportate in calce, per riferimento).

```
typedef long Align;

union header {
    struct {
        union header *ptr;
        unsigned size;
    } s;
    Align x;
};

typedef union header Header;

static Header base;
static Header *freep = NULL;

void *malloc(unsigned nbytes)
{
    Header *p, *prevp;
    unsigned nunits;

    nunits=(nbytes+sizeof(Header)-1)/sizeof(Header)+1;
    if((prevp = freep) == NULL) {
        base.s.ptr = freep = prevp = &base;
        base.s.size = 0;
    }
    for(p = prevp->s.ptr; ; prevp = p, p = p->s.ptr) {
        if(p->s.size >= nunits) {
            if(p->s.size == nunits)
                prevp->s.ptr = p->s.ptr;
            else {
                p->s.size -= nunits;
                p += p->s.size;
                p->s.size = nunits;
            }
            freep = prevp;
            return (void *)(p + 1);
        }
        if(p == freep)
            if((p = sbrk(nunits)) == NULL)
                return NULL;
    }
}
```

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 memory 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

Register Usage

Name	Register Num.	Usage
$zero	0	The constant value 0
$s0-$s7	16-23	Saved
$t0-$t9	8-15,24-25	Temporaires
$a0-$a3	4-7	Arguments

Name	Register Num.	Usage
$v0-$v1	2-3	Results
$fp, $sp	30,29	frame pointer, stack pointer
$ra, $gp	31,28	return address, global pointer
$k0-$k1	26,27	Kernel usage

Name	Usage
$f0, $f1, … , $f31	Single precision floating point registers
$f0, $f2, … , $f30	Double precision floating point registers

System calls

Service Name	Service Num. ($v0)	INPUT Arguments	OUTPUT Arguments
print_int	1	$a0=integer to print	---
print_float	2	$f12=float to print	---
print_string	4	$a0=address of ASCIIZ string to print	---
sbrk	9	$a0=Number of bytes to be allocated	$v0=pointer to the allocated memory
exit	10	---	---

2) [6/40] Con riferimento ad una struttura con pipeline a 5 stadi MIPS si indichi nel seguente codice quali tipi di criticita' (hazards) sono presenti:

```
add $1, $2, $3
lw   $2, 0($1 )
sub $1, $1, $2
```