COG	NOMS:																											
	NOM:]											
exam tacho	ORTAN nen. Esc ones ni o cuadro	criba bor	a un rone	solo es y	o ca que	rácto cad	er po la ca	or re rácte	cua er q	dro, uede	en r e en	nay mar	úsc cac	ulas do de	y lo ntro	más de	cla su r	ram ecu	ente adro	pos sin	ible lleg	e. Es ar a	im to	porta car lo	nte s bo	que rdes	no h . Use	aya un
Prob	olema :	1. (2	2,5	pun	tos)																						
Un pi en ei	rogram ecutars	a P	ejec x10 ⁹	uta :	5x10	0 ⁹ in en u	stru n pr	ccior	nes ado	diná r C c	mica aue f	as y unc	rea ion	liza 3 ia a u	x10 na f	9 op	erac ienc	cion	es d	e pu GHz.	nto	flot	ant	e. El p	orogr	rama	Pta	rda
_	Calcula						-				-																	
El nr		0.5.4					ri o n.		- f		Jo 11	0. 4		alim				٠٠١٤	oio «			. +ic					m a ali	tivo
-	ocesad valente									_								volt	aje (de 2	Vy	' tie	ene	una	carg	a ca	pacı	tiva
b)	Calcula	a la	ene	rgía	con	sum	ida	al eje	ecut	tar e	l pro	gra	ma	P en	el p	roce	esac	dor	С									
-	ograma							_	-	-				-														
c)	Calcula progra			nerc	(N)) de	pro	cesa	dor	es id	lénti	cos	a C	C que	sor	n ne	cesa	rios	s pai	а сс	nse	guii	r uı	n spe	ed-u	p de	4 e	n el
d)	Calcula	a los	MF	LOF	S de	е Р є	n di	cho	siste	ema	para	alelo	1) c	l pro	cesa	dore	es C).										
	n sisten del sis	-					-								-			r tie	ene	un N	/ITTF	: de	<u>:</u> 10) millo	ones	de l	noras	s. El
e)	Calcula	a el	tien	ро	med	dio h	asta	falle	o (N	/ITTF) de	l sist	ten	na co	mpl	eto.												

COGNOMS:														
NOM:														

Problema 2. (2.5 puntos)

Dado el siguiente código escrito en C que compilamos para un sistema linux de 32 bits:

```
typedef struct {
    short a[3];
    char b[3];
    short c[2];
    short c[2];
} s1;
typedef struct {
    shu[2];
    int v;
    char w[5];
    char *x;
} s2;
```

a)	Dibuja como quedarían almacenadas en memoria las estructuras s1 y s2 , indicando claramente los desplazamientos respecto al inicio, el tamaño de todos los campos y el tamaño de los structs.
b)	Escribe UNA ÚNICA INSTRUCCIÓN que permita mover z.u[1].c[0] al registro %ax, siendo z una variable de tipo s2 cuya dirección está almacenada en el registro %ebx. Indica claramente la expresión aritmética utilizada para e cálculo de la dirección.
1	

COGNOMS:														
NOM:														

Problema 3. (2.5 puntos)

Dado el siguiente código escrito en C:

```
int examen(short int a, short int b, short int v[10], int M[10][10]) {
  int ii;
  short int aa, bb;
  short int vector[10];
  int *matriz;
  ...
  return M[ii][ii];
}
```

a)	Dibuja el bloque de activación de la rutina examen, indicando claramente los desplazamientos respecto a %ebp y
	el tamaño de todos los campos.

b)	Traduce a ensamblador del x86 la instrucción return M[ii][ii] suponiendo que no hay que restaurar el valor de ningún registro.

COGNOMS:														
NOM:														

Problema 4. (2.5 puntos)

Cuando se va a ejecutar la siguiente instrucción escrita en ensamblador del x86:

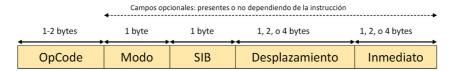
```
movl $337, -40(%ebp, %esi, 4)
```

jns bucle

El contenido de los registros del procesador (en hexadecimal) es:

```
eax: 0x00000005 ebx: 0x00000004 ecx: 0x00000001 edx: 0x00000002 esp: 0x80800f30 ebp: 0x80800f48 esi: 0x00000002 edi: 0x00001001 eip: 0x808000fe eflags: 0x286
```

Sabemos que la codificación de instrucciones sigue el siguiente esquema tal y como se explica en clase:



Además sabemos que el procesador tiene una cache de datos de 32Kbytes (copy back+write allocate), y otra de instrucciones de 16Kbytes, ambas con lineas de 32 bytes y asociatividad 2. Sus páginas son de 4Kbytes y dispone de un TLB de 4 entradas. Asumimos que tanto el TLB como las caches están vacíos.

a)	¿Cuantos bytes ocuparía la instrucción anterior si sabemos que el OpCode de movl solo ocupa 1 byte?
L	
<u>b)</u>	¿Cuantos bloques de memoria se leen para ejecutar el movl? Escribe la dirección de dichos bloques.
c)	¿Cuantos bloques de memoria se escriben para ejecutar el movl? Escribe la dirección de dichos bloques.
d)	¿Cuantos fallos de TLB tendremos al ejecutar la instrucción anterior? Indica a que páginas es el fallo
Sup	oniendo el siguiente código:
	leal N-1(), %esi <- Esta instrucción ocupa 2 bytes
	bucle: movl \$337, -40(%ebp, %esi, 4) <- Instrucción a evaluar
	decl %esi

e) ¿Cuantos fallos de cache provoca el movl en función del valor de N? (suponemos de nuevo que las caches están vacías y los registros tienen valor indicado anteriormente antes de ejecutar el código).

<- El bucle hace N iteraciones