2º Grado Informática
Estructura de Computadores
9 Septiembre 2013

Examen Test (3.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.
Cada respuesta vale 3/30 si es correcta, 0 si está en blanco o claramente tachada, -1/30 si es errónea.
Anotar las respuestas (a, b, c o d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
c	c	b	d	b	d	b	b	d	c	c	a	a	a	c	a	a	b	c	b	b	d	a	b	d	a	c	b	a	b

Examen de Prácticas (4.0p)

Todas las preguntas son de elección simple sobre 4 alternativas.
Cada respuesta vale 4/20 si es correcta, 0 si está en blanco o claramente tachada, -4/60 si es errónea.
Anotar las respuestas (a, b, c o d) en la siguiente tabla.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
a	a	b	a	b	c	a	b	a	a	b	d	b	a	b	a	c	a	c	b

Examen de Problemas (3.0p)

1. **Bucles for** (0.5 puntos).

A.

```
long fun_c(unsigned long x) {
  long val = 0;
  int i;
  for (i = 0; i < 8; i++) {
     val += x & 0x0101010101010101;
     x >>= 1;
  }
  val += (val >> 32);
  val += (val >> 16);
  val += (val >> 8);
  return val & 0xFF;
}
```

B. Calcula el nº bits a 1 (popcount) de x.
Va haciendo 8 sumas parciales en paralelo en los 8 bytes de val.
Luego se suman las 4 sumas superiores (7-4) con las 4 inferiores (3-0),
se siguen acumulando 2 superiores (3-2) con 2 inferiores (1-0),
y por último, la suma del byte (1) con la del byte inferior (0).
Se retorna sólo ese byte, en donde están sumados todos los bits.
La ventaja es que hace sólo 8 iteraciones, 3 sumas y una máscara
en lugar de 64 iteraciones.

2. Representación y acceso a estructuras (0.5 puntos).

(A)	a	b	c	d	e	f	g	h	total (B)
tamaño	4	2	8	1	4	1	8	4	(4B relleno)
desplaz	0	4	8	16	20	24	32	40	48
(C)	**c**	**g**	**a**	**e**	**h**	**b**	**d**	**f**	**total (C)**
tamaño	8	8	4	4	4	2	1	1	(sin relleno)
desplaz	0	8	16	20	24	28	30	31	32

3. Entrada/Salida (0.5 puntos).
Se podrían usar 2 (N)ANDs para decodificar: E/S 0x220 activa IO/M, A9 y A5, 0x221 activa además A0
Drivers triestado E/S (E obligatorio) desde-hacia terminales Q-D de los registros (tipo D, 8bits)
Segundo nivel (N)AND según R/W para CLK-Load Salida o driver Entrada (driver Salida activado decodificador)
Se espera que temporización R/W ⊂ IO/M-Addr, para que CLK-Load pulse con datos estables

4. Diseño del sistema de memoria (0.5 puntos).
SRAM: módulos 4Kx8 → 12bits Addr, 8bits Data
espacio 0x0-0x1fff=13bits → 8K
8Kx32 = (4Kx8) x (2x4) → 2x4 = 8 módulos
ROM: módulos 8Kx16 → 13bits Addr, 16bits Data
espacio 0xc000-0xffff como 0x0-3fff=14bits → 16K
16Kx32 = (8Kx16) x (2x2) → 2x2 = 4 módulos
Dibujar: SRAM: dos hileras de 4 módulos, 0x0-0xfff y 0x1000-0x1fff,
datos D0-7, D8-15, D16-23, D24-31, direcciones A0-11, a todos los módulos
dirs A12-31 para decodificadores NAND (todas invertidas 1ª hilera, A12 no invertida 2ª hilera)
ROM: dos hileras de 2 módulos, 0xc000-0xdfff y 0xe000-0xffff,
datos D0-15, D16-31, direcciones A0-12, a todos los módulos, A13-31 para decods NAND
1ª hilera todas invertidas salvo A15-14 (0xc*,0xd*), 2ª hilera A15-13 (0xe*, 0xf*)
R/W a todas las SRAM (R/W) y ROM (OE) (R/W invertida si OE activa baja)

5. Memoria cache (0.5 puntos).
A. Asociativa: 32 = 24 + 8. etiqueta +palabra.
B. Directa: 32 = 16 + 8 + 8. etiqueta+marco+palabra. 2^{16} B/cach / 2^8 B/bloq = 2^8 bloq/cache
C. Asociativa Conjuntos 16vías: 32 = 20 + 4 + 8. etiqueta+conjnt+palabra. 2^8 bloq/cach / 2^4 blq/conj = 2^4 conj/cache

6. Unidad de control (0.5 puntos).

			Detallar señales de control no se pide	
FETCH:	MAR := PC,	Z := PC+4	(EN_{PC}, LD_{MAR}),	(EN_{PC}, $MUX2_4$, ALU_1, LD_Z)
	Read,	PC := Z	(ciclo lectura),	(EN_Z, LD_{PC})
	MBR := Mem		($MUX1_{MEM}$, LD_{MBR})	
	IR := MBR		(L_{IR}, $EN_{MBR-BUS}$)	
	goto f(IR)		No sabemos cómo obtener codop de IR, cómo va goto, suponer f(IR)	
ADDR:	Y := Rsrc		(EN_{Rsrc}, LD_Y) obtener src-dst de IR? suponer que vale decir Rsrc	
	Z := Y + Rdst		($MUX2_Y$, ALU_1, EN_{Rdst}, LD_Z)	
	Rdst := Z,	goto FETCH	(EN_Z, LD_{Rdst}) No sabemos cómo va goto, suponer OK en paralelo	
ADDI:	MAR := Rsrc		(EN_{Rsrc}, LD_{MAR})	
ADDI_rd:	Read			
	MBR := Mem,	Y := Rdst	($MUX1_{MEM}$, LD_{MBR}),	(EN_{Rdst}, LD_Y)
	Z := Y + MBR		($MUX2_Y$, ALU_1, $EN_{MBR-BUS}$, LD_Z)	
	Rdst := Z,	goto FETCH	(EN_Z, LD_{Rdst})	
ADDX:	MAR := PC,	Z := PC+4	(EN_{PC}, LD_{MAR}),	(EN_{PC}, $MUX2_4$, ALU_1, LD_Z)
	Read,	PC := Z	(como fetch),	(EN_Z, LD_{PC})
	MBR := Mem,	Y := Rsrc	($MUX1_{MEM}$, LD_{MBR}),	(EN_{Rsrc}, LD_Y)
	Z := Y + MBR		($MUX2_Y$, ALU_1, $EN_{MBR-BUS}$, LD_Z)	
	MAR := Z,	goto ADDI_rd	(EN_Z, LD_{MAR}), resto como addi	