COGNOMS: **GRUP:**

NOM:

EXAMEN PARCIAL D'EC1
Dijous, 19 d'abril de 2007

L'examen consta de 7 preguntes. S'ha de contestar als mateixos fulls de l'enunciat, dins dels requadres. No oblideu posar el vostre nom i cognoms a tots els fulls. La duració de l'examen és de **120 minuts**. Les notes, la data de revisió i la solució es publicaran el dia 3 de maig al Racó.

Pregunta 1. (2 punts)

Donades les següents declaracions de variables globals i de subrutines, en C:

```
# define N 10
int mat[N][N];

void subr1 (int *p, int i)
{
     *p = subr2(&mat[i-3][i+1], N);
}

int subr2 (int *a, int b)
{
     register int aux = *a;
     return ((aux * b) + 255);
}
```

a) Quins són els registres que `subr1` ha de preservar-ne el valor obligatòriament pel fet de cridar a `subr2`?

R1 i R6

b) Tradueix a llenguatge assemblador SISA-F la subrutina `subr1`

```
$PUSH   R1,R6
MOVI    R0,N+1
MUL     R0,R2,R0            ; R0= i*(N+1)
MOVI    R3,-3*N+1
ADD     R0,R0,R3            ; R0= i*(N+1) - 3*N+1
ADD     R0,R0,R0
$MOVEI  R3,mat
ADD     R1,R0,R3            ; R1= @mat + (i*(N+1)-3*N+1)*2
MOVI    R2,N
$CALL   R6,subr2
$POP    R6,R1
ST      0(R1),R0
JMP     R6
```

c) Tradueix a llenguatge assemblador SISA-F la subrutina `subr2`

```
LD      R0, 0(R1)           ; R0= aux = *a
MUL     R0,R0,R2            ; R0= aux*b
$MOVEI  R1, 255
ADD     R0,R0,R1            ; R0= aux*b + 255
JMP     R6
```

Pregunta 2. (1,50 punts)

a) Donat el número `N=0x43C9` escrit en coma flotant SISA-F, converteix-lo al format IEEE-754 de simple precisió, sabent que aquest format consta d'1 bit de signe, 8 d'exponent (en excés a 127) i 23 de mantissa (amb bit ocult). Dóna el resultat en hexadecimal:

N = **0x 40F24000**

(8 dígits hexadecimals)

b) Sabent que els valors inicials dels registres F2 i F3 del SISP-F són `0x4406` i `0x43D8` respectivament, ¿quin és el contingut del registre F1, en hexadecimal, després d'executar la instrucció `SUBF F1,F2,F3` ?

F1 = **0x 3B40**

(4 dígits hexadecimals)

COGNOMS: **GRUP:**

NOM:

Pregunta 3. (1,25 punts)

Suposem que denotem el valor inicial de cada bit de R1 amb una lletra de la següent manera (per ex., n és el valor del bit 2):

R1 = **abcd efgh ijkl mnop**

Seguint la mateixa notació, escriu el contingut del registre R0 després d'executar les següents instruccions:

```
        MOVI    R0,0
        $MOVEI  R2,0x4000
        MOVI    R3,-7
        MOVI    R4,0
for:
        MOVI    R5,8
        $CMPLT  R6,R4,R5
        BZ      R6,fifor
        AND     R6,R1,R2
        SHL     R6,R6,R3
        OR      R0,R0,R6
        MOVI    R6,-2
        SHL     R2,R2,R6
        ADDI    R3,R3,1
        ADDI    R4,R4,1
        BNZ     R4,for
fifor:
        HALT
```

R0 = **0000 0000 bdfh jlnp**

Pregunta 4. (2,25 punts)

Donada la següent declaració de dades en C:

```
struct examen {
        long    num;
        char    car;
        int     vec[10];
        float   real;
};

struct examen tupla;
struct examen *ptupla;
long *plong;
```

a) Indica el desplaçament en bytes que s'ha de sumar a l'adreça inicial d'una tupla de tipus "examen" per accedir a l'inici de cada un dels seus camps:

Desplaçament a "num": **0** Desplaçament a "car": **4**

Desplaçament a "vec": **6** Desplaçament a "real": **26**

Indica també la mida total en bytes que ocupen en memòria les següents variables:

Mida de "tupla": **28** Mida de "ptupla": **2** Mida de "plong": **2**

b) Tradueix el següent codi en C a llenguatge assemblador SISA-F:

```
if ((tupla.car == 'A') || (tupla.vec[0] == 0))
     plong = &tupla.num;
else
     tupla.real = 0.0;
```

```
      $movei   r1, tupla
      ldb      r2, 4(r1)            ; llegim tupla.car
      movi     r3, 'A
      $cmpeq   r2, r2, r3           ; tupla.car == 'A'?
      bnz      r2, if
      ld       r2, 6(r1)            ; llegim tupla.vec[0]
      bnz      r2, else             ; tupla.vec[0] == 0?
 if:
      $movei   r3, plong
      st       0(r3), r1            ; plong = &tupla.num
      bnz      r3, end
 else:
      subf     f1, f1, f1           ; f1 <- 0.0
      stf      26(r1), f1           ; tupla.real = 0.0
 end:
```

c) Tradueix el següent codi en C a llenguatge assemblador SISA-F:

```
*plong = ptupla->num;
```

```
      $movei   r1, ptupla
      ld       r1, 0(r1)            ; r1 <- adreça a on apunta ptupla
      ld       r2, 0(r1)            ; r2 <- word baix del camp num
      ld       r3, 2(r1)            ; r3 <- word alt del camp num
      $movei   r1, plong
      ld       r1, 0(r1)            ; r1 <- adreça a on apunta plong
      st       0(r1), r2            ; escrivim el word baix
      st       2(r1), r3            ; escrivim el word alt
```

d) Tradueix el següent codi en C a llenguatge assemblador SISA-F:

```
ptupla = ptupla + 3;
```

```
      $movei   r1, ptupla
      ld       r2, 0(r1)            ; r2 <- valor de ptupla
      movi     r3, 3 * 28           ; r3 <- 3 * mida tupla examen
      add      r2, r2, r3           ; r2 <- ptupla + 3 * 28 (aritmètica punters)
      st       0(r1), r2            ; escrivim ptupla
```

COGNOMS: **GRUP:**

NOM:

Pregunta 5. (1 punt)

El següent codi en C modifica N elements d'una matriu d'enters:

```
#define N 10
int matriu[N][N];
main()
{
     register int i;              // i ocupa el registre R1
     for (i=0; i<N; i++)
     {
             matriu[N-i-1][N-i-1]++;
     }
}
```

Tradueix el codi anterior a llenguatge assemblador SISA-F utilitzant la tècnica d'accés seqüencial. Recorda que "i" ocupa el registre R1.

```
; Calculem l'adreça del primer element a recórrer:
; @matriu[N-1][N-1] = matriu + (N-1)*N*2 + (N-1)*2 = matriu + N*N*2 - 2

     $movei  r2, matriu + N*N*2 - 2; inicialitzem punter = @matriu[N-1][N-1]
     movi    r1, 0                  ; i=0
     movi    r3, N
for:
     $cmplt  r4, r1, r3             ; i<N?
     bz      r4, fifor
     ld      r5, 0(r2)              ; llegim element usant el punter
     addi    r5, r5, 1              ; incrementem el valor
     st      0(r2), r5              ; guardem valor usant el punter
     addi    r2, r2, -(N+1)*2       ; sumem el stride al punter
     addi    r1, r1, 1              ; i++
     bnz     r4, for
fifor:
```

Pregunta 6. (1 punt)

Completa els espais (1) amb una de les operacions: "XOR"/"AND"/"OR", i els espais (2) amb un número hexadecimal. Nota: els codis ASCII de 'A', 'a' i '0' són, respectivament 65, 97 i 48.

a) Suposant que R1 conté el codi ASCII d'una lletra, per convertir-la a majúscules cal fer l'operació ..... R1=R1 **AND** 0x**DF (o 0x5F)**

b) Suposant que R1 conté el codi ASCII d'una lletra, per convertir-la a minúscules cal fer l'operació ..... R1=R1 **OR** 0x**20**

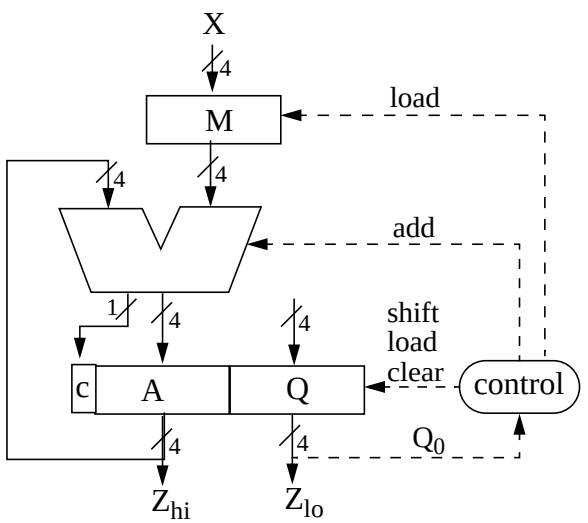
c) Suposant que R1 conté el codi ASCII d'una lletra, per intercanviar majúscules per minúscules i viceversa cal fer l'operació ... R1=R1 **XOR** 0x**20**

d) Suposant que R1 conté el codi ASCII d'un dígit numèric, per calcular el valor binari representat per R1 cal fer l'operació .... R1=R1 **AND** 0x**0F (o 0x4F, 0x8F, 0xCF)**

(1) (2)

Pregunta 7. (1 punt)

Sigui un multiplicador seqüencial de dos números naturals Z=X*Y, com el que has estudiat a classe, però amb operands de 4 bits en comptes de 16, i resultat de 8 bits en comptes de 32:




```
// Inicialitzar els registres
M  =  X;
Q  = Y;
CA = 0;
// Algorisme amb 4 iteracions
for (i=1; i<=4; i++)
{
    if (Q0 == 1)
        CA = A + M;
    CAQ = CAQ >> 1;
}
// Aquests són els resultats
Zlo = Q;
Zhi = A;
```

a) Completa l'algorisme en alt nivell que apareix al costat del diagrama de blocs, el qual descriu el funcionament del multiplicador. Pots referir-te als valors continguts en els registres C, A i Q com a "CA", "AQ", o "CAQ", segons et convingui. Considera que tots els registres de l'algorisme són números naturals, i per tant els operadors en alt nivell >> i << expressen desplaçaments lògics.

b) Volem multiplicar els números binaris `X=1110` i `Y=1101` amb l'anterior multiplicador. Es demana que escriguis, <u>en hexadecimal, a la columna de la dreta</u> de la següent taula, el contingut del registre AQ (8 bits) just després d'inicialitzar-lo i just després de cada un dels 4 passos de multiplicació. Per obtenir aquests resultats en hexadecimal, és necessari que primer omplis les altres caselles buides, amb els valors en binari dels registres C, A i Q.

	M	C	A			Q					AQ (hex)
Després d'inicialitzar-los	1 1 1 0	0	0 0 0 0			1 1 0 1				=	0x0D
Després de la 1ª iteració		0	0 1 1 1			0 1 1 0				=	0x76
Després de la 2ª iteració		0	0 0 1 1			1 0 1 1				=	0x3B
Després de la 3ª iteració		0	1 0 0 0			1 1 0 1				=	0x8D
Després de la 4ª iteració		0	1 0 1 1			0 1 1 0				=	0xB6