COGNOMS: **GRUP:**

NOM:

EXAMEN PARCIAL D'EC1
Divendres, 9 de novembre de 2007

L'examen consta de 8 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 durada de l'examen és de **120 minuts**. Les notes i la solució es publicaran el dia 20 de novembre al Racó, i la revisió serà l'endemà dia 21.

Pregunta 1. (1,50 punts)

Donades les següents declaracions de variables globals, en C:

```
struct tuplaQ {
    int    x;
    char   ent8[6];
    char   car;
};

struct tuplaQ var1 = {-1, {0x80, 0x81, 0x82, 0x83, 0x84, 0x85}, 0x62};
int var2 = -2;
char *var3 = &var1.ent8[2];
char var4 = -116;
long var5 = 0x00A3B805;
```

a) Tradueix les anteriors declaracions a SISA-F:

```
.data
  var1: .word -1
        .byte 0x80,0x81,0x82,0x83,0x84,0x85
        .byte 0x62
        .balign 2
  var2: .word -2
  var3: .word var1+2+2*1
  var4: .byte -116
        .balign 2
  var5: .long 0x00A3B805
```

b) Suposant que aquestes variables s'ubiquen a partir de l'adreça 0x100 de memòria, completa la següent taula amb el contingut de la memòria byte per byte, en hexadecimal. Les posicions sense inicialitzar han de contenir el valor 00. El format de la taula és anàleg al que mostra el simulador *sisa-dbg* del laboratori.

Adreça		Contingut de memòria (en hexadecimal)							
0x100	:	FF	FF	80	81	82	83	84	85
0x108	:	62	00	FE	FF	04	01	8C	00
0x110	:	05	B8	A3	00	00	00	00	00

c) Quin és el valor final de R1 després d'executar el següent fragment de codi, tenint en compte les inicialitzacions de variables de les declaracions de l'enunciat?

```
$MOVEI  R3,var5
LD      R1,2(R3)
$MOVEI  R3,var3
LD      R2,0(R3)
STB     -1(R2),R1
LD      R1,-2(R2)
```

R1 = **0xA380**

Pregunta 2. (0,75 punts)

Suposant que els registres R1 i R2 contenen nombres enters de 16 bits, completa el fragment de codi següent amb les instruccions necessàries (no més de 6 instruccions) per tal que el valor final de R3 sigui 1 en cas que la multiplicació `MUL R0,R1,R2` produeixi desbordament (el producte d'enters no és representable amb 16 bits); i que R3 valgui 0 en cas contrari.

```
MUL R0,R1,R2
MOVI R5,-15
SHA R5,R0,R5
MULH R3,R1,R2
$CMPNE R3,R3,R5
```

COGNOMS: **GRUP:**

NOM:

Pregunta 3. (1,50 punts)

Donades les següents declaracions d'accions i funcions:

```
int funcio1(int *a, char b) { ... };
void accio1(char a1, char *a2) { ... };
int subrutina(int sv[10], char sc[10], int i)
{
     register int res_func1;    // res_func1 es guarda a R4
     register char *p_c;        // p_c es guarda a R5

     res_func1 = funcio1(&sv[3], sc[i]);
     p_c = sc + res_func1;
     accio1(*p_c, p_c);
     return sv[res_func1];
}
```

a) Indica quins registres s'han de salvar a la pila obligatòriament a la funció *subrutina*() pel fet de cridar a *accio1*().

Registres = R6,R1,R4

b) Tradueix a llenguatge assemblador SISA-F el codi de la funció *subrutina*(). Recorda que les variables *res_func1* i *p_c* ocupen el registres R4 i R5 respectivament.

```
subrutina:
     $PUSH R1,R2,R6
     ADDi R1,R1, 3*2 ; &sv[3] (par1)
     ADD  R2,R2,R3   ; &sc[i]
     LDB  R2,0(R2)   ; sc[i]  (par2)
     $MOVEI R6,funcio1
     JAL R6,R6
     $POP R6,R2,R1
     ADDi R4,R0,0    ;res_func1=...

     ADD  R5,R2,R4   ;p_c=sc+ ...

     $PUSH R1,R4,R6
     LDB  R1,0(R5)   ;*p_c (par1)
     ADDi R2, R5,0   ; p_c (par2)
     $MOVEI R6,accio1
     JAL R6,R6
     $POP R6,R4,R1

     ADD R1,R1,R4
     ADD R1,R1,R4   ; &sv[res_func1]
     LD  R0,0(R1)
     JMP R6
```

Pregunta 4. (0,75 punts)

Donat el següent fragment de codi en SISA-F, i suposant que R1 conté un número enter en complement a 2, indica amb una [X] dins el corxet quina de les següents afirmacions és certa (sols 1):

```
MOVI    R2,-15
SHA     R2,R1,R2
XOR     R0,R1,R2
SUB     R2,R0,R2
```

a) [ ] El valor final de R2 és igual a: `(R1 xor -1) - (-1)`

b) [ ] El valor final de R2 és igual a: `(not R1) - 1`

c) [ ] El valor final de R2 és igual a R1

d) [**X**] El valor final de R2 és igual al valor absolut de R1

e) [ ] El valor final de R2 és igual al doble de R1

f) [ ] Cap de les anteriors és certa

COGNOMS: **GRUP:**

NOM:

Pregunta 5. (2 punts)

Donades les següents declaracions de variables globals, en C:

```
char a, b;
char *p;
```

Tradueix la següent sentència a llenguatge SISA-F

```
if ((a>b) || (b!=0))
    p = &a;
else
    *p = *p + b;
```

```
; solució 1

    $MOVEI  R0,a
    LDB     R1,0(R0)
    $MOVEI  R2,b
    LDB     R2,0(R2)
    $MOVEI  R4,p
    $CMPGT  R3,R1,R2
    BNZ     R3,then
    BZ      R2,else
then:
    ST      0(R4),R0
    MOVI    R0,0
    BZ      R0,final
else:
    LD      R5,0(R4)
    LDB     R3,0(R5)
    ADD     R3,R3,R2
    STB     0(R5),R3
final:
```

```
; solució 2

    $MOVEI  R0,a
    LDB     R1,0(R0)
    $MOVEI  R2,b
    LDB     R2,0(R2)
    $MOVEI  R4,p
    $CMPGT  R3,R1,R2
    OR      R3,R3,R2
    BZ      R3,else

    ST      0(R4),R0
    BNZ     R3,final

else:
    LD      R5,0(R4)
    LDB     R3,0(R5)
    ADD     R3,R3,R2
    STB     0(R5),R3
final:
```

Pregunta 6. (2 punts)

Donat el següent codi en C:

```
int mat[N][M];

main()
{
     register int i;            // i es guarda en R0
     register int producte;     // producte es guarda en R1

    producte = 1;
     for (i=N-1; i>=0; i-=3)
            producte = producte * mat[i][N-1-i];
}
```

Tradueix el codi del *main* a llenguatge assemblador SISA-F utilitzant la **tècnica d'accés seqüencial**. Recorda que les variables "i" i "producte" ocupen els registres R0 i R1 respectivament

```
main:
     ; Punter R2 = @mat[N-1][0] = mat+(N-1)*M*2
     ; Stride R5 = @mat[i-3][N-1-i+3] - @mat[i][N-1-i] = -(M-1)*3*2

     MOVI    R1,1               ; producte=1
     $MOVEI  R0,N-1             ; i=N-1
     $MOVEI  R2,mat+(N-1)*M*2   ; inicialitzem punter
     $MOVEI  R5,-(M-1)*3*2      ; inicialitzem stride
for:
     MOVI    R4,0
     $CMPGE  R4,R0,R4           ; i>=0 ?
     BZ      R4,fifor
     LD      R3,0(R2)           ; carreguem mat[i][N-1-i]
     MUL     R1,R1,R3           ; producte = producte*mat[i][N-1-i]
     ADD     R2,R2,R5           ; increment (negatiu) del punter
     ADDI    R0,R0,-3           ; i-=3
     BNZ     R4,for
fifor:
     HALT
```

COGNOMS: **GRUP:**

NOM:

Pregunta 7. (1 punts)

Suposant que F1 i F2 contenen els valors 0x43D3 i 0xB840 respectivament, quin és el valor de F3 en hexadecimal després d'executar la instrucció: `ADDF F3,F1,F2` ?

R1 = **0x43C1**

Observeu que, com que F2 és negatiu, **restem les mantisses** en lloc de sumar-les.

Pregunta 8. (0,50 punts)

El següent codi en SISA-F fa la **suma de naturals** dels valors guardats a les variables "a" i "b", de tipus long, i deixa el resultat a la variable "res":

```
1    .data
2            a: .long VALOR_A
3            b: .long VALOR_B
4            res: .long 0

5    .text
6    main:
7            $MOVEI R0,a
8            $MOVEI R1,b
9            $MOVEI R5,res
10           LD R2,0(R0)
11           LD R3,0(R1)
12           ADD R2,R2,R3
13           ST 0(R5),R2
14           $CMPLTU R4,R2,R3
15           LD R2,2(R0)
16           LD R3,2(R1)
17           ADD R2,R2,R3
18           ADD R2,R2,R4
19           ST 2(R5),R2
20           HALT
```

Indica quantes línies caldria modificar i quines serien les modificacions, per tal que el codi faci la **suma d'enters** de les variables "a" i "b" de tipus long.

No cal modificar **cap línia**.

La suma de números naturals i la d'enters en complement a 2 segueixen el mateix algorisme i donen el mateix resultat. Aquesta és precisament una de les principals propietats de la codificació en complement a 2.