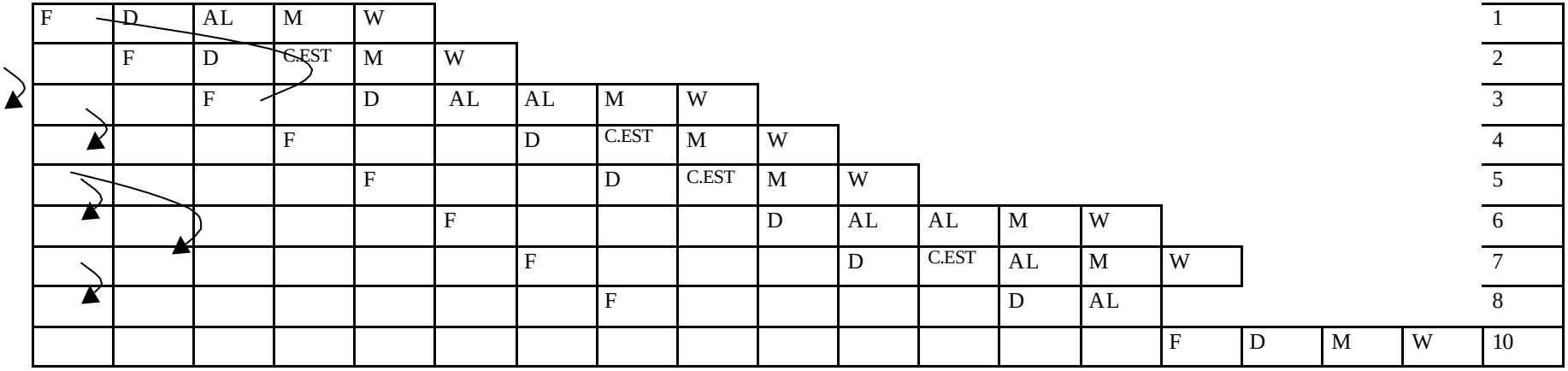
EXERCICI 1:

A)



B) Les dependències que trobem son les següents:

- RAW entre la instrucció 3 i 1 (r3); entre la instrucció 2 i 1 (r2); entre la instrucció 3 i 4 (r4); entre la instrucció 5 i 6 (r5); entre la instrucció 5 i 7 (r5); entre la instrucció 7 i 8 (r7)

- Dependència estructurals entre la instrucció 3 i 4 (memòria); entre la instrucció 4 i 5 (memòria); entre la instrucció 6 i 7 (alu).

c)

```
1          const    r3, base_a          ;
           nop
2          load     r2, (r1)            ;
3          add      r4, r2, r3
           nop
           nop
4          store    r4, (r2)            ;
5          load     r5, (r3)
6          add      r6, r5, r4          ;
           nop
7          cmplt    r7, r5, r4          ;
           nop
8          jmp      r7, salt            ;
9          br       fi                  ;
           nop
           nop
10         store    r7, (r1)            ;
11 fi:
```

EXERCICI 2:

	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
MULT S1,S2,S3	F	D	L	X	X	X	X	W																		
ADD S7,S1,S3	F	D							L	X	X	W														
SUB S2,S7,S5		F	D										L	X	X	W										
CMP S6,S7,0		F	D										L	X	W											
BNE S6,LL1			F	D												L	X									
ADD S5,S5,S2			F	D													L									
MULT S3,S4,S1				F	D				L	X	X	X	X	W												
SUB S4,S4,S3				F									D		L	X	X									
LL1: DIV S5,S7,S5													F					F	D	L	X	X	X	X	W	
ADD S4,S3,S1													F					F	D	L	X	X	W			

Estacions de reserva i banc de registres el cicle:8

NOM	OCUP	OPER	FI	FJ	QJ	FK	QK	RJ	RK
+1	2	+	S7	S1	(*1)	S3		NO	SI
+2	3	-	S2	S7	(+1)	S5		NO	SI
+3	6	+	S5	S5		S2	(+2)	SI	NO
*1	1	*	S1	S2		S3		SI	SI
*2	7	*	S3	S4		S1		SI	SI
SEN1	4	CMP	S6	S7	(+1)	0		NO	SI
SEN2	5	BNE		S6	(S1)			NO	SI

	R1	R2	R3	R4	R5	R6	R7
Vi							
Qi	(*1)	(+2)	(*2)		(+3)	(S1)	(+1)

Estacions de reserves i banc de registres els cicles:17

NOM	OCUP	OPER	FI	FJ	QJ	FK	QK	RJ	RK
+1	8	-	S4	S4		S3		SI	SI
+2									
+3	6	+	S5	S5		S2		SI	SI
*1									
*2									
SEN1									
SEN2	5	BNE		S6				SI	

	R1	R2	R3	R4	R5	R6	R7
Vi	(*1)	(+2)	(*2)			S1	(+1)
Qi				(+1)	(+3)		

Estacions de reserva i banc de registres els cicle:12

NOM	OCUP	OPER	FI	FJ	QJ	FK	QK	RJ	RK
+1	2	+	S7	S1		S3		SI	SI
+2	3	-	S2	S7	(+1)	S5		NO	SI
+3	6	+	S5	S5		S2	(+2)	SI	NO
*1									
*2	7	*	S3	S4		S1		SI	SI
SEN1	4	CMP	S6	S7	(+1)	0		NO	SI
SEN2	5	BNE		S6	(S1)			NO	SI

	R1	R2	R3	R4	R5	R6	R7
Vi	(*1)						
Qi		(+2)	(*2)		(+3)	(S1)	(+1)

Estacions de reserva i banc de registres els cicles:20

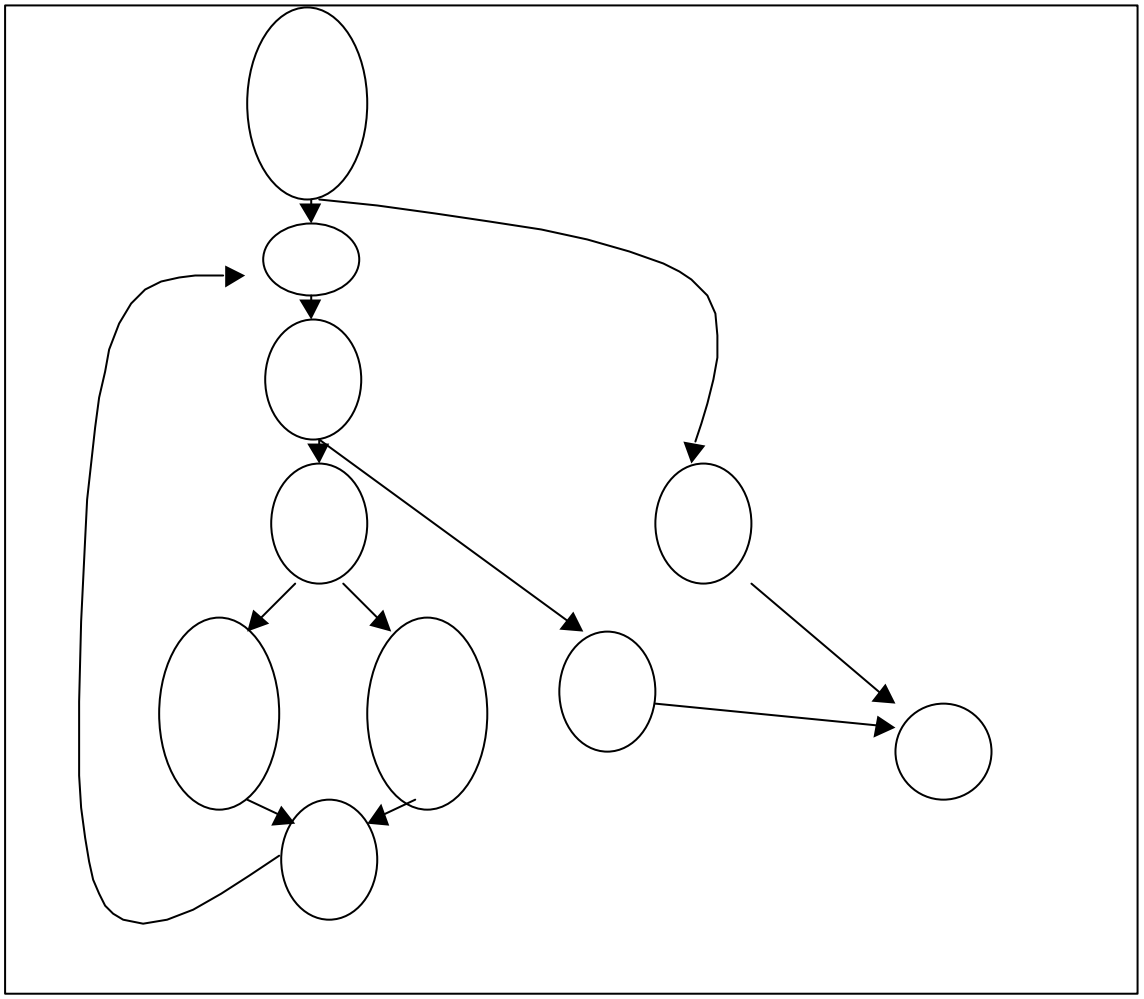
NOM	OCUP	OPER	FI	FJ	QJ	FK	QK	RJ	RK
+1	10	+	S5	S7		S5		SI	SI
+2									
+3									
*1	9	/	S4	S3		S1		SI	SI
*2									
SEN1									
SEN2									

	R1	R2	R3	R4	R5	R6	R7
Vi							
Qi				(*1)	(+1)		

b) Els registres es trobaran abans del salt en estat arquitectònic (instruccions acabades en ordre i desordre) i després es trobaran en estat d'ordre donat a que s'ha salvat el contingut.

EXERCICI 3:

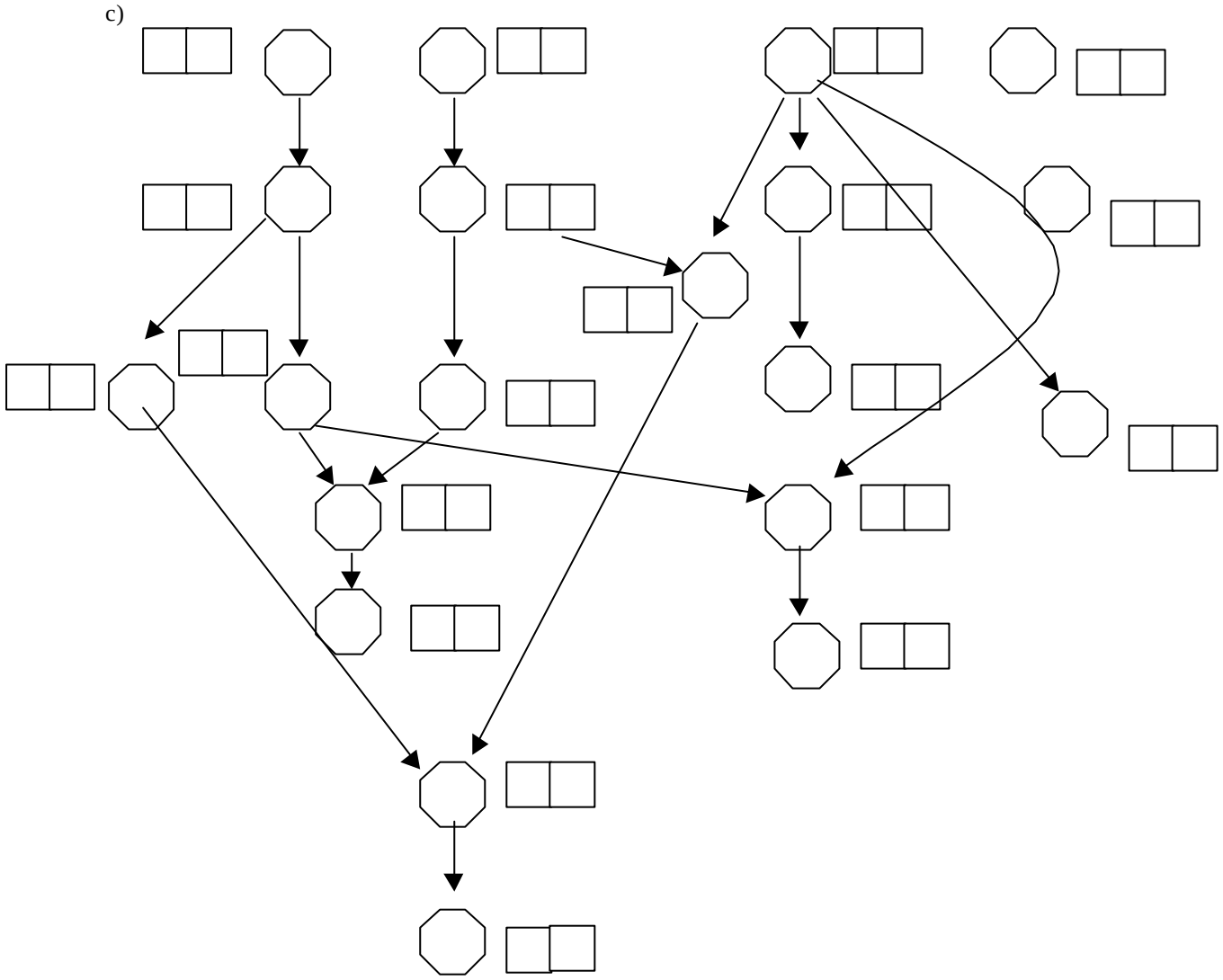
a)



b) APLICANT EL LOOP UNROLLING LA SOLUCIO ÉS LA SEGÜENT:

```
1          mov r20 var_1
2          mov r21 var_2
3          load r0,(r20)0
4          load r1,(r21)0
5          sub r2,r0,1
6          mult r3,r1,2
7          cmp r4,r2,r3
8          bne r4,LL3
9          mov r17,0
10    LL1: cmp r18,r17,5
11         bne r18,LL5
12         cmp r19,r2,r17
13         ble r19,LL6
14         add r10,r5,r0
15         mult r11,r1,r17
16         mult r12,r10,r11
17         store r12, (r22)0
18         b LL7
19    LL6: sub r13,r5,r0
20         div r14,r1,r0
21         add r15,r13,r17
22         store r15,(r20)0
23    LL7: add r17,r17,1
24         cmp r19,r2,r17
25         ble r19,LL9
26         add r10,r5,r0
27         mult r11,r1,r17
28         mult r12,r10,r11
29         store r12, (r22)0
30         b LL17
31    LL9: sub r13,r5,r0
32         div r14,r1,r0
33         add r15,r13,r17
34     store r15,(r20)0
35   LL17: add r17,r17,1
36         b LL1
37    LL5: b LL2
38    LL3: sub r16,r0,r1
39         store r16,(r21)0
40    LL2:
```

c)



aTaula de reserva

cicle	Llista	UF:MEM	UF:COMP	UF:PF	UF:SEN	UF:SAL	UF:COMP
1	2(0),9(1),1(2),18(10),24(10)		2(0)			18(10)	9(1)
2	4(1),1(2),10(9),24(10),23(10)	4(1)	10(9)		23(10)	24(10)	1(2)
3	6(5),15(2),3(3),11(10)	3(3)		15(2)		11(10)	
4	14(4),5(7),6(5)				14(4)		
5	16(6),6(5),5(7)			6(5)	5(7)		
6	16(6),7(9),12(9)						
7	16(6),7(9),12(9)		12(9)	16(6)			
8	7(9),17(10),13(10)				13(10)		
9	7(9),17(10)		7(9)				
10	8(10),17(10)					8(10)	
11	17(10)			17(10)			

d) L'aplicació del trace schedulling en un bucle comporta els problemas de que instruccions de fora el bucle poden entrar a dins del mateix i quan es dona més d'una iteració això complicaria molt l'execució del bucle pel codi de restauració que s'hauria d'introduir. Per tant la solució seria aplicar el trace schedulling només de les instruccions que pertanyen al bucle.

EXERCICI 4

Disposem del següent codi :

```
for(j=1;j<1000;j++)
        {
                for(i=1;i<1000;i++)
                {
                        for (k=1;k<1000;k++)
                        {
                                c[i][j] = SUB (a[i][k],b[k][j]);
                                t = SUB (c[i][j],a[i]);
                        }
```

Indicar com afectaria l'aplicació de les següents tècniques d'optimització. Justificar quines es podem aplicar i quines no, i quines podrien arribar a millorar el rendiment. Al efectuar un perfex, tenim el següent perfil en les caches:

% perfex -e 25 multmat <- Per obtenir el número de falles de cahcé de L1

0 Cycles.....59002735555
25 Primary data caché misses.....1338524786

% perfex -e 26 multmat <- Per obtenir les falles de caché de L2

0 Cycles.....59018761168
26 Secondary data caché misses.....54515279

Temps d'execució 351 segons

Tècnica d'optimització	Justificació
SW pipelining	No és pot aplicar ja que tenim una dependència de dades al bucle.
I. Cache Blocking	OK pot millorar el rendiment. Especialment amb mides de bloc grans.
II. Loop fusion	No podem aplicar-la ja que no tenim bucles per fusionar.
III. Inlining	OK. Podem aplicar-la sobre la funció resta (SUBB).
IV. Loop Interchange	OK. Pot funcionar ja que tenim un bucle amb índexs que podem accedir de forma ineficient a les dades.

EXERCICI 5

Respon, justificant la resposta, les següents preguntes sobre PVM :

a) Quina instrucció fem servir per inicialitzar un buffer (canal) per enviar dades. De forma que les dades es codifiquin en un format estándar a dues arquitectures diferents ?. Com s'anomena el protocol que efectua la conversió de dades per tal de fer-la transparent al programador ?

Solució :

pvm_initsend (PvmDataDefault);
XDR

b) Quin missatge d'error hauria d'aparèixer al 'printf' en el següent fragment de codi PVM:

```
int SON[25];
pvm_spawn("sontask", NULL, PvmTaskDefault, 0, 2, SON);

if (pvm_send(SON[1], 2) < 0) {
    printf("????? \n");
    pvm_exit();
```

Solució:

Problemes en la tasca PARE al enviar el buffer al FILL 2

c) Quin missatge d'error hauria d'aparèixer al 'printf' en el següent fragment de codi PVM:

```
if(pvm_recv( parent_tid, 2) < 0) {
    printf("?????, fin");
    pvm_exit();
    return -1;
```

Solució:

Error al rebre dades del pare.