Si consideri un sistema costituito dal processore D-RISC sequenziale (quello dell'interprete firmware) e dotato di una gerarchia di memoria che comprende un'unico livello di cache, comune per dati e istruzioni, set associativa su insiemi (4 linee per insieme), con σ = 16 e 1K insiemi. La memoria principale è modulare, interallacciata, con 4 moduli da 1G parole ciascuno e τ_M=400 τ (con τ ciclo di clock della CPU). Il t_{tr} fra memoria principale e cache è pari a 4τ.
Si consideri quindi il codice D-RISC che deriva dalla compilazione dello pseudo codice

```
int N = 1024, A[N], B[N], C[N];
for(int i=0; i<N; i++)
{
        if(A[i] == B[i]) C[i] = A[i]*4;
        else C[i] = A[B[i]%N]*2;
}
```

e si calcolino:

- la compilazione in assembler DRISC
- il numero di fault generati dal programma
- la traccia degli indirizzi generati dal programma e quelli utilizzati per l'accesso alla cache durante la prima iterazione del for, indicando per ciascun indirizzo il numero di insieme nella cache
- il tempo di completamento della prima iterazione, assumendo che A[0] sia diverso da B[0]

assumendo che:

- i vettori A, B e C e il codice siano allocati rispettivamente dal compilatore agli indirizzi 1K, 2K, 3K e 0
- che le pagine in memoria centrale e le pagine logiche siano di 2K parole
- che tabril[] = {<12,1>,<6,1>,<24,1>,<1,1>,…} dove ogni coppia è data da <IPF, bit di presenza>

Commenti:

- ➔ per semplicità, si assume N potenza di 2 (N=1024)
- ➔ Compilazione del codice -> standard. C'è da ricordarsi che la moltiplicazione e l'operazione di modulo sono per potenze di due dunque vanno fatte con gli shift.

Codice

La compilazione con le regole standard produce il codice:

```
        CLEAR Ri  (ADD R0,R0,Ri)
loop:   LOAD Rba, Ri, Rai
        LOAD Rbb, Ri, Rbi
        IF= Rai, Rbi, then
else:   AND Rbi, #1023, Rind  // operazione modulo 1024: 1023=0...01111111111
                              // quindi si ottengono gli ultimi 10 bit di Rbi
        LOAD Rba, Rind, Rai
        SHL Rai, #1, Rai  // = *2: quel valore di Rai non si userà più e si può modificare
        STORE Rbc, Ri, Rai
        GOTO cont
then:   SHL Rai, #2, Rai  //= *4: quel valore di Rai non si userà più e si può modificare
        STORE Rbc, Ri, Rai
cont:   INC Ri
        IF< Ri, RN, loop
        END
```

Numero di fault

L'accesso ai vettori è:

per B e C sequenziale, senza riuso,
per A random o sequenziale.

Utilizzando il flag "prefetch" per **B** e **C** abbiamo il solo fault iniziale (quindi **1+1**: senza prefetch sarebbero $N/\sigma+N/\sigma$),
mentre per **A** dovremmo prevedere comunque **N/σ** faults, per l' istruzione ***A[B[i] %N]*** (caso peggiore).

Però, se si verificasse un fault per un accesso ad A non ancora toccato dagli accessi sequenziali per il test della condizione (cioè se ***B[i]%N>i***), il fault non verrebbe ripetuto quando si arriva a tale linea di cache per l'esecuzione di IF=,
perchè la capacità complessiva della cache (1K insiemi da 4 linee contenenti 16 parole =4K linee=64K parole) è sufficiente a contenere tutto il vettore A, oltre alla singola linea di B e C nel working set, ed al codice.
Ma non è il caso pessimo e non lo consideriamo.

Il **codice**, dopo il fault iniziale (cioè **1**), non ne produrrà altri, utilizzando il prefetch (sono **14** istruzioni).

Il tempo di completamento deve tener conto di #fault * T_{fault}. Quest'ultimo sarà dato dalla solita formula $\mathbf{2(\tau + T_{tr}) + \sigma\, t_M / m}$.

Quindi, in totale il numero dei faults è:

$$3+N/\sigma = 67$$

Traccia degli indirizzi

Gli indirizzi generati saranno quelli tradotti dalla **tabril**: entra tutto in due pagine = 4K parole (ne usiamo 3K+14) => si usano solo le pagine **IPF = 12** e **IPF = 6**.

La traccia degli indirizzi logici generati dal programma per la prima iterazione è la seguente:

0, 1, 1024, 2, 2048, 3,

cioè, in termini di <IPL, OFF>:

<0,0>, <0,1>, <0,1024>, <0,2>, <1,0>, <0,3>

e prosegue, in caso il test **IF= abbia successo**, con

9, 10, 3072, 11, 12

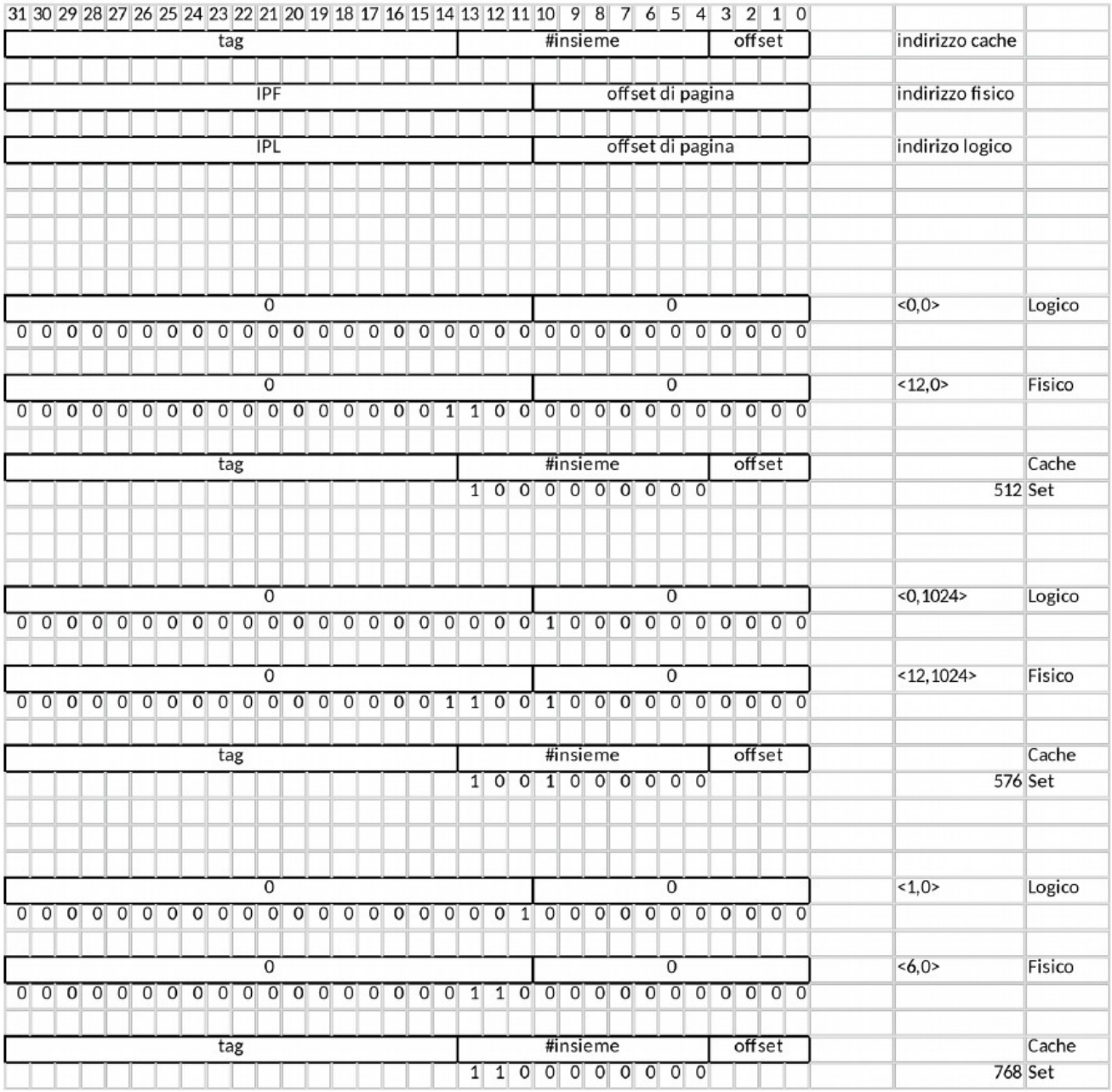
<0,9>, <0,10>, <1, 1024>, <0,11>, <0,12>

else, con

4, 5, X (fra 1024 e 2047 compresi), 6, 7, 3072, 8, 11, 12

<0,4>, <0,5>, <0,X>, <0,6>, <0,7>, <1,1024>, <0,8>, <0,11>, <0,12>

Gli indirizzi fisici corrispondono ad indirizzi logici ricavati mediante la tabella di rilocazione. Le **pagine** sono da **2K** parole, quindi **codice** e vettore **A** sono nella **prima** pagina logica e quindi nella pagina fisica con **IPF = 12** (come da tabril), mentre i vettori **B** e **C** sono nella seconda pagina logica e vengono mappati nella pagina fisica con **IPF = 6**.



la traccia degli **indirizzi fisici** generati verso la cache dalla MMU sarà (considerando solo la parte iniziale per semplicità):
set=512, offset=0, tag=1
set=512, offset=0, tag=1
set=576, offset=0, tag=0
…
gli accessi in cache considerano i 32 bit dell'indirizzo fisico come (dai bit meno significativi a quelli più significativi):

- **4 bit di offset**
- **10 bit di numero di blocco**
- **18 bit di tag**

Tempo di completamento prima iterazione

La prima iterazione esegue (ramo else) 3 LOAD, 1 STORE, 2 IF, 1 GOTO e 3 aritmetico logiche corte. Complessivamente fanno **37τ+14ta**:

		Texec		CH0, CH1		Totale	
	#	τ	t_a	τ	t_a	τ	t_a
LOAD	3	2	1	2	1	12	6
STORE	1	3	1	2	1	5	2
SALTI COND	2	2	0	2	1	8	2
SALTI INCOND	1	1	0	2	1	3	1
ARIT LOG CORTE	3	1	0	2	1	9	3
				Tot		37	14

+ i tempi necessari per i **fault** relativi al **codice**, ad **A[0]** a **B[0]**, ad **A[B[i]%N]** (probabilmente non risolto con la stessa linea che contiene A[0]) e (in sola scrittura), a **C[0]** (trascurabile rispetto agli altri), che in **totale** costano

$$4*(2(\tau+ T_{tr}) + (\sigma\ \tau_M)/m)=4*(2(\tau+ 4\tau)+(16*400\tau)/4)=4(10\tau+1600\tau) =$$
$$=4*1610\ \tau = 6440\ \tau$$

e quindi il tempo di completamento della **prima iterazione** sarà

$$37\tau+14ta + 6440\ \tau$$

con **ta**=tempo di accesso in **cache** di primo livello, sarà di **2 o 3 τ** a seconda del modo scelto per implementare la cache set associativa (**1 o 2 τ + 1 τ per la MMU**), il **tempo complessivo** sarà (**cache** set associativa con accesso in **1 τ+ 1 τ per la MMU**)

$$6440\ \tau + 14\ (2\ \tau) + 37\ \tau = 6505\ \tau$$