```
int a[16];          // 4 bytes per int
in negatius=0;
...
for (i=0; i<15; i++)
   if (a[i]<0) {
      negatius += a[i];
      a[i]=0;
      }
```

a[16]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	+	+	+	+	+	-	-	+	+	+	+	+	-	-	-	-
memory line (block)	line 0				line1				line2				line3			

Cache size = 48 bytes
Line size = 16 bytes
Random placement, random replacement, copy-back, write-allocate

line data D/V: dirty/valid bits

line	data	D/V
0		
1		
2		

miss ratio (a) = 4/22
miss ratio (negatius) = 1/12

P	a[16]					negatius (line 4)		
	element	memory line	cache line	D/V bits	hit/miss	cache line	D/V bits	hit/miss
0	0	0	0	0/1	m (r)		0/1	
0	1	0	0	0/1	h (r)		0/1	
0	2	0	0	0/1	h (r)		0/1	
0	3	0	0	0/1	h (r)		0/1	
0	4	1	1	0/1	m (r)		0/1	
0	5	1	1	0/1 -> 1/1	h (r), h (w)	2	0/1 -> 1/1	m (r), h (w)
0	6	1	1	1/1	h (r), h (w)	2	1/1	h (r), h (w)
0	7	1	1	1/1	h (r)	2	1/1	
0	8	2	1 (*)	0/1	m (r)	2	1/1	
0	9	2	1	0/1	h (r)	2	1/1	
0	10	2	1	0/1	h (r)	2	1/1	
0	11	2	1	0/1	h (r)	2	1/1	
0	12	3	0 (**)	0/1 -> 1/1	m (r), h (w)	2	1/1	h (r), h (w)
0	13	3	0	1/1	h (r), h (w)	2	1/1	h (r), h (w)
0	14	3	0	1/1	h (r), h (w)	2	1/1	h (r), h (w)
0	15	3	0	1/1	h (r), h (w)	2	1/1	h (r), h (w)

(*) random replacement of line 1, implies flush to MP because line dirty

```
int a[16];          // 4 bytes per int
in negatius=0;
...
#pragma omp taskloop gainsize(2)
for (i=0; i<15; i++)
   if (a[i]<0) {
      #pragma omp atomic
      negatius += a[i];
      a[i]=0;
      }
```

a[16]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	+	+	+	+	+	-	-	+	+	+	+	+	-	-	-	-
memory line (block)	line 0				line1				line2				line3			
task assigned to processor ...	1		0		0		1		1		0		0		1	

Cache size=48 bytes
Line size = 16 bytes
Random placement, random replacement, copy-back, write-allocate
Coherence protocol: write-invalidate MSI

cache 0

line	data	MSI
0		
1		
2		

miss ratio (a) = 6/11
miss ratio (negatius) = 2/6

cache 1

line	data	MSI
0		
1		
2		

miss ratio (a) = 6/11
miss ratio (negatius) = 3/6

Note: Observe that miss ratio is worse than single processor!

P	a[16]							negatius (line 4)				
	element	memory line	cache 0/1	cache line	Bus event	State (MSI)	hit/miss	cache 0/1	cache line	Bus event	State (MSI)	hit/miss
1	0	0	1	0	BusRd1	S for 1	m (r)					
0	2	0	0	0	BusRd0	S for 0	m (r)					
1	1	0	1	0		no change	h (r)					
0	3	0	0	0		no change	h (r)					
0	4	1	0	1	BusRd0	S for 0	m (r)					
1	6	1	1	1	BusRd1 BusRdX1 (inval)	S then I for 0 S then M for 1	m (r), h (w)	1	2	BusRd1 BusRdX1	S then M for 1	m (r), h (w)
0	5	1	0	1	BusRd0, Flush1 BusRdX0 (inval)	S then M for 0 S then I for 1	m (r), h (w)	0	2	BusRd0, Flush1 BusRdX0 (inval)	S then M for 0 S then I for 1	m (r), h (w)
1	7	1	1	1	BusRd1, Flush0	S for 0, S for 1	m (r)				no change	
1	8	2	1	1 (*)	BusRd1	S for 1	m (r)				no change	
0	10	2	0	1	BusRd0	S for 0	m (r)				no change	
1	9	2	1	1		no change	h (r)				no change	
0	11	2	0	1		no change	h (r)				no change	
0	12	3	0	0 (**)	BusRd0 BusRdX0 (inval)	S then M for 0	m (r), h (w)	0	2		no change	h (r), h (w)
1	14	3	1	0	BusRd1, Flush0 BusRdX1 (inval)	S then I for 0 S then M for 1	m (r), h (w)	1	2	BusRd1, Flush0 BusRdX1 (inval)	S then I for 0 S then M for 1	m (r), h (w)
0	13	3	0	0	BusRd0, Flush1 BusRdX0 (inval)	S then M for 0 S then I for 1	m (r), h (w)	0	2	BusRd0, Flush1 BusRdX0 (inval)	S then M for 0 S then I for 1	m (r), h (w)
1	15	3	1	0	BusRd1, Flush0 BusRdX1 (inval)	S then I for 0 S then M for 1	m (r), h (w)	1	2	BusRd1, Flush0 BusRdX1 (inval)	S then I for 0 S then M for 1	m (r), h (w)

(*) random replacement of line 1, does not imply flush to MP since already updated
(**) random replacement of line 0: does not imply flush