Emisión desordenada 4i/c

MOV → 2c LD → 2c MULT → 4c

Add → 1c ST → 2c

SUB → 1c Lógicas → 2c Sin PRED

1. Ejecutamos en un superescalar segmentado

Nº	Instrucción	Explicación	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	movss (%r12), %xmm0	xmm0 ← M[r12]	F	D	Ex	Ex	W									
2	subss %xmm2, %xmm1	xmm1 ← xmm1-xmm2	F	D	Ex		W									
3	addq $4, %rbp	rbp ← rbp + 4	F	D	Ex		W									
4	addq $4, %r12	r12 ← r12+4	F	D		Ex	W									
5	subss %xmm2, %xmm0	xmm0 ← xmm0-xmm2		F	D		Ex	W								
6	divss %xmm7, %xmm1	xmm1 ← xmm1 / xmm7		F	D	Ex	Ex	Ex	Ex	W						
7	divss %xmm7, %xmm0	xmm0 ← xmm0 / xmm7		F	D			Ex	Ex	Ex	Ex	W				
8	movss %xmm1, -4(%rbp)	M[rbp-4] ← xmm1		F	D					Ex	Ex	W				
9	movaps %xmm0, %xmm1	xmm1 ← xmm0			F	D						Ex	Ex	W		
10	movss %xmm0, -4(%r12)	M[r12-4] ← xmm0			F	D							Ex	Ex	W	
11	pxor %xmm0, %xmm0	xmm0 ← xmm0 xor xmm0			F	D						Ex	Ex		W	
12	subss %xmm4, %xmm1	xmm1 ← xmm1-xmm4			F	D								Ex	W	
13	mulss %xmm1, %xmm1	xmm1 ← xmm1*xmm1				F	D								Ex	Ex Ex Ex W Ex
14	addss %xmm1, %xmm3	xmm3 ← xmm3+xmm1				F	D									

Nº	Instrucción	Explicación	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	movss (%r12), %xmm0	xmm0 ← M[r12]	1	1												
2	subss %xmm2, %xmm1	xmm1 ← xmm1-xmm2	11	11												
3	addq $4, %rbp	rbp ← rbp + 4	1	1												
4	addq $4, %r12	r12 ← r12+4	1	1	1											
5	subss %xmm2, %xmm0	xmm0 ← xmm0-xmm2	10	10	10	11										
6	divss %xmm7, %xmm1	xmm1 ← xmm1 / xmm7	10	10	11											
7	divss %xmm7, %xmm0	xmm0 ← xmm0 / xmm7	10	10	10	10	11									
8	movss %xmm1, -4(%rbp)	M[rbp-4] ← xmm1	00	00	01	01	01	01	11							
9	movaps %xmm0, %xmm1	xmm1 ← xmm0	0	0	0	0	0	0	0	0	1					
10	movss %xmm0, -4(%r12)	M[r12-4] ← xmm0	00	00	00	01	01	01	01	01	11	11				
11	pxor %xmm0, %xmm0	xmm0 ← xmm0 xor xmm0	00	00	00	00	00	00	00	00	11					
12	subss %xmm4, %xmm1	xmm1 ← xmm1-xmm4	10	10	10	10	10	10	10	10	10	10	11			
13	mulss %xmm1, %xmm1	xmm1 ← xmm1*xmm1	00	00	00	00	00	00	00	00	00	00	00	11		
14	addss %xmm1, %xmm3	xmm3 ← xmm3+xmm1	01	01	01	01	01	01	01	01	01	01	01	01	01	01 01 11

Código para el ejercicio de predicción de saltos de la Convocatoria Ordinaria

Código de Alto Nivel	Código ensamblador simplificado

...
int i = 0;
while(i<N){
  if(a[i]%7==0) break;
  i++;
}
	.Salto_while: … ; Dirección de Salto_while
	...
	...
	je .Salto_if Salto positivo
	…
	...
	jle .Salto_while Salto negativo
	.Salto_if: … ; Dirección de Salto_if, fuera del bucle while

Tabla de Saltos condicionales

Dirección Salto	Dirección objetivo Salto	Estado
je .Salto_if	Salto_if	101
jle .Salto_while	Salto_while	000

N = 9

a = 4 1,3,5,9,11,10,8,6,2 9

i=0 i=1 i=2 i=3 i=4 i=5 i=6 i=7 i=8 i=9 i=10

	1	2	3	4	5	6	7	8	9	10	...					
Ejecucion Salto_If	N	N	N	N	N	N	N	N	N	N						
Historial Salto_If	101	010	101	010	101	010	101	010	101	010						
Predicción Salto If	S	N	N	N	N	N	N	N	N	N						
Ejecución Salto_While	S	S	S	S	S	S	S	S	S	S	N					
Historial Salto_While	000	100	010	110	111	111	111	111	111	111	111					
Predicción Salto_While	N	N	S	S	S	S	S	S	S	S	S					

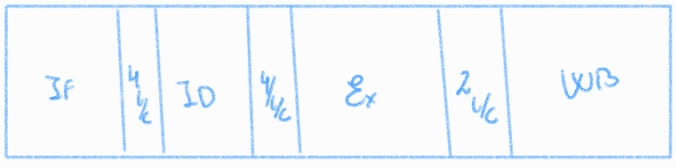
N=9 a[i]%7≠0 ✓

Tabla de Saltos condicionales

Dirección Salto	Dirección objetivo Salto	Estado
je .Salto_if	Salto_if	01
jle .Salto_while	Salto_while	00

i=9

	1	2	3	4	5	6	7	8	9	10	...					
Ejecucion Salto_If	N	N	N	N	N	N	N	N	N	N						
Historial Salto_If	01	00	00	00	00	00	00	00	00	00						
Predicción Salto If	N	N	N	N	N	N	N	N	N	N						
Ejecución Salto_While	S	S	S	S	S	S	S	S	S	S	N					
Historial Salto_While	00	01	10	11	11	11	11	11	11	11	11					
Predicción Salto_While	N	N	S	S	S	S	S	S	S	S	S					



Emisión desordenada

Nº	Instrucción	Explicación	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	movss (%r12), %xmm0	xmm0←M[r12]	f	Ex	Ex	Ex	RAB	W								
2	subss %xmm2, %xmm1	xmm1 ← xmm1-xmm2	f	Ex	RAB			W								
3	addq $4, %rbp	rbp← rbp + 4	f	Ex	RAB			W								
4	addq $4, %r12	r12 ← r12+4	f		Ex	AB		W								
5	subss %xmm2, %xmm0	xmm0 ← xmm0-xmm2		f			Ex	AB	W							
6	divss %xmm7, %xmm1	xmm1← xmm1 / xmm7		f	Ex	Ex	Ex	AB	W							
7	divss %xmm7, %xmm0	xmm0← xmm0 / xmm7			f			Ex	Ex	Ex	AB	W				
8	movss %xmm1, -4(%rbp)	M[rbp-4] ← xmm1			f			Ex	Ex	AB		W				
9	movaps %xmm0, %xmm1	xmm1 ← xmm0				f					Ex	Ex	AB	W		
10	movss %xmm0, -4(%r12)	M[r12-4] ← xmm0				f							Ex	Ex	AB	W
11	pxor %xmm0, %xmm0	xmm0 ← xmm0 xor xmm0				f					Ex	AB				W
12	subss %xmm4, %xmm1	xmm1← xmm1-xmm4				f							Ex	AB		W
13	mulss %xmm1, %xmm1	xmm1 ← xmm1*xmm1												Ex	Ex	Ex AB W
14	addss %xmm1, %xmm3	xmm3 ← xmm3+xmm1														Ex AB W

Nº	Instrucción	Explicación	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	movss (%r12), %xmm0	xmm0←M[r12]	1													
2	subss %xmm2, %xmm1	xmm1 ← xmm1-xmm2	11													
3	addq $4, %rbp	rbp← rbp + 4	1													
4	addq $4, %r12	r12 ← r12+4	1	1												
5	subss %xmm2, %xmm0	xmm0 ← xmm0-xmm2	10	10	10	11										
6	divss %xmm7, %xmm1	xmm1← xmm1 / xmm7	10	11												
7	divss %xmm7, %xmm0	xmm0← xmm0 / xmm7	10	10	10	10	11									
8	movss %xmm1, -4(%rbp)	M[rbp-4] ← xmm1	00	01	01	01	11									
9	movaps %xmm0, %xmm1	xmm1 ← xmm0	0	0	0	0	0	0	0	1						
10	movss %xmm0, -4(%r12)	M[r12-4] ← xmm0	00	00	01	01	01	01	01	11	11	11				
11	pxor %xmm0, %xmm0	xmm0 ← xmm0 xor xmm0	00	00	00	00	00	00	00	11						
12	subss %xmm4, %xmm1	xmm1← xmm1-xmm4	10	10	10	10	10	10	10	10	10	11				
13	mulss %xmm1, %xmm1	xmm1 ← xmm1*xmm1	00	00	00	00	00	00	00	00	00	00	11			
14	addss %xmm1, %xmm3	xmm3 ← xmm3+xmm1	01	01	01	01	01	01	01	01	01	01	01	01	01	11

Emisión ordenada segmentada con todos menos mulf y div a 1c

Nº	Instrucción	Explicación	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	movss (%r12), %xmm0	xmm0 ← M[r12]	F	D	Ex	W										
2	subss %xmm2, %xmm1	xmm1 ← xmm1-xmm2	F	D	Ex	W										
3	addq $4, %rbp	rbp ← rbp + 4	F	D	Ex	W										
4	addq $4, %r12	r12 ← r12+4	F	D		Ex	W									
5	subss %xmm2, %xmm0	xmm0 ← xmm0-xmm2		F	D	Ex	W									
6	divss %xmm7, %xmm1	xmm1 ← xmm1 / xmm7		F	D	Ex	Ex	W								
7	divss %xmm7, %xmm0	xmm0 ← xmm0 / xmm7		F	D		Ex	Ex	W							
8	movss %xmm1, -4(%rbp)	M[rbp-4] ← xmm1		F	D			Ex	W							
9	movaps %xmm0, %xmm1	xmm1 ← xmm0			F	D			Ex	W						
10	movss %xmm0, -4(%r12)	M[r12-4] ← xmm0			F	D				Ex	W					
11	pxor %xmm0, %xmm0	xmm0 ← xmm0 xor xmm0			F	D			Ex		W					
12	subss %xmm4, %xmm1	xmm1 ← xmm1-xmm4			F	D				Ex	W					
13	mulss %xmm1, %xmm1	xmm1 ← xmm1*xmm1				F	D				Ex	Ex	W			
14	addss %xmm1, %xmm3	xmm3 ← xmm3+xmm1				F	D						Ex	W		

Nº	Instrucción	Explicación	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	movss (%r12), %xmm0	xmm0 ← M[r12]	1	1	/											
2	subss %xmm2, %xmm1	xmm1 ← xmm1-xmm2	11	11	/											
3	addq $4, %rbp	rbp ← rbp + 4	1	1	/											
4	addq $4, %r12	r12 ← r12+4	1	1	1	/										
5	subss %xmm2, %xmm0	xmm0 ← xmm0-xmm2	10	10	11	/										
6	divss %xmm7, %xmm1	xmm1 ← xmm1 / xmm7	10	10	11	/										
7	divss %xmm7, %xmm0	xmm0 ← xmm0 / xmm7	10	10	10	11	/									
8	movss %xmm1, -4(%rbp)	M[rbp-4] ← xmm1	00	00	01	01	11	/								
9	movaps %xmm0, %xmm1	xmm1 ← xmm0	0	0	0	0	0	1	/							
10	movss %xmm0, -4(%r12)	M[r12-4] ← xmm0	00	00	00	01	01	11	11	/						
11	pxor %xmm0, %xmm0	xmm0 ← xmm0 xor xmm0	00	00	00	00	00	11	/							
12	subss %xmm4, %xmm1	xmm1 ← xmm1-xmm4	10	10	10	10	10	10	11	/						
13	mulss %xmm1, %xmm1	xmm1 ← xmm1*xmm1	00	00	00	00	00	00	00	11	/					
14	addss %xmm1, %xmm3	xmm3 ← xmm3+xmm1	01	01	01	01	01	01	01	01	01	11	/			