



Presentación Práctica 3C – ASD Límites ILP

- Pedro Escobar Rubio
- Alejandro Fernández Trigo

Índice

- Detalles de ejecución
- Problema tratado
- Códigos elaborados
 - Test (duración, pruebas, ensamblador)
 - Flujo de datos, camino crítico, ILP
 - Otros
- Bibliografía

Máquinas donde se ejecuta el código

Máquina 1

Intel Core i7-10750H Comet Lake

Frecuencia: 4489.02 MHz.

Caché: L1 Data: 6 x 32KB.

L1 Inst.: 6 x 32KB.

Level 2: 6 x 256KB.

Level 3: 12MB.

Memoria: DDR4 16GB.

Máquina 2

Intel Core i7-6500U Skylake-U/Y

Frecuencia: 2990.40 MHz.

Caché: L1 Data: 2 x 32KB.

L1 Inst.: 2 x 32KB.

Level 2: 2 x 256KB.

Level 3: 4MB.

Memoria: DDR4 4GB.

Enunciado

Sumatorio de los elementos de un vector. Probar con:

- a. Una sola suma por iteración.
- b. Dos sumas por iteración.
- c. Una suma condicional por iteración:
 - i. La condición es difícilmente predecible.
 - ii. La condición es fácilmente predecible por la BTB.

Bucles – Ejemplo provisto

```
float example ()
{
    int i;

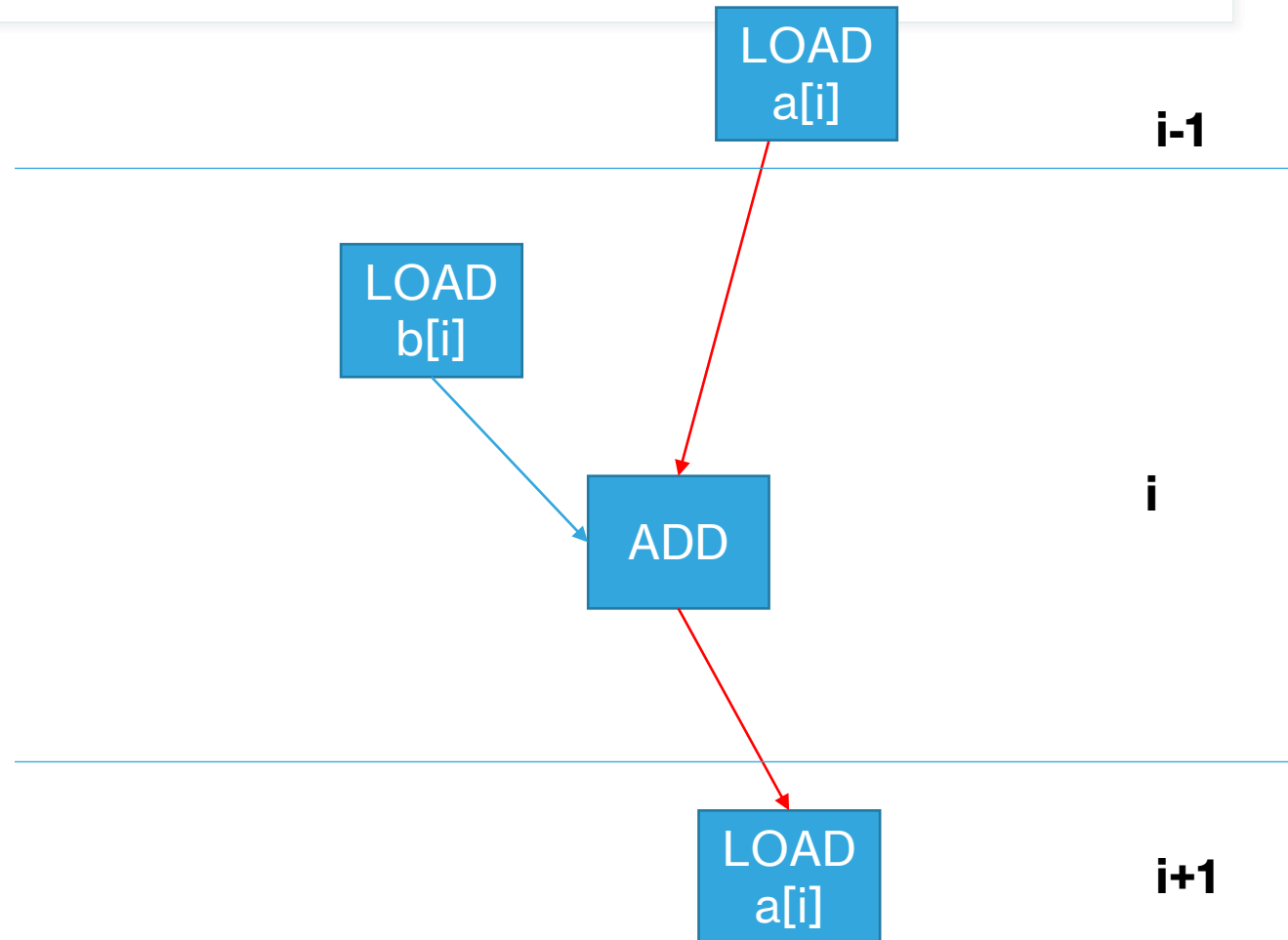
    for (i=0;i<N_ELEM;i++)
    {
        a[i]=b[i]+8;
    }

    return a[N_ELEM-1];
}
```

```
00961330 movss xmm0,dword ptr b (09653C0h)[eax]
00961338 addss xmm0,xmm1
0096133C movss dword ptr a (096D3C0h)[eax],xmm0
00961344 movss xmm0,dword ptr [eax+9653C4h]
0096134C addss xmm0,xmm1
00961350 movss dword ptr [eax+96D3C4h],xmm0
00961358 movss xmm0,dword ptr [eax+9653C8h]
00961360 addss xmm0,xmm1
00961364 movss dword ptr [eax+96D3C8h],xmm0
0096136C movss xmm0,dword ptr [eax+9653CCh]
00961374 addss xmm0,xmm1
00961378 movss dword ptr [eax+96D3CCh],xmm0
00961380 movss xmm0,dword ptr [eax+9653D0h]
00961388 addss xmm0,xmm1
0096138C movss dword ptr [eax+96D3D0h],xmm0
00961394 movss xmm0,dword ptr [eax+9653D4h]
0096139C addss xmm0,xmm1
009613A0 movss dword ptr [eax+96D3D4h],xmm0
009613A8 movss xmm0,dword ptr [eax+9653D8h]
009613B0 addss xmm0,xmm1
009613B4 movss dword ptr [eax+96D3D8h],xmm0
009613BC movss xmm0,dword ptr [eax+9653DCh]
009613C4 addss xmm0,xmm1
009613C8 movss dword ptr [eax+96D3DCh],xmm0
009613D0 add eax,20h
009613D3 cmp eax,2000h
009613D8 jl example+10h (0961330h)
```

Bucles – Ejemplo provisto

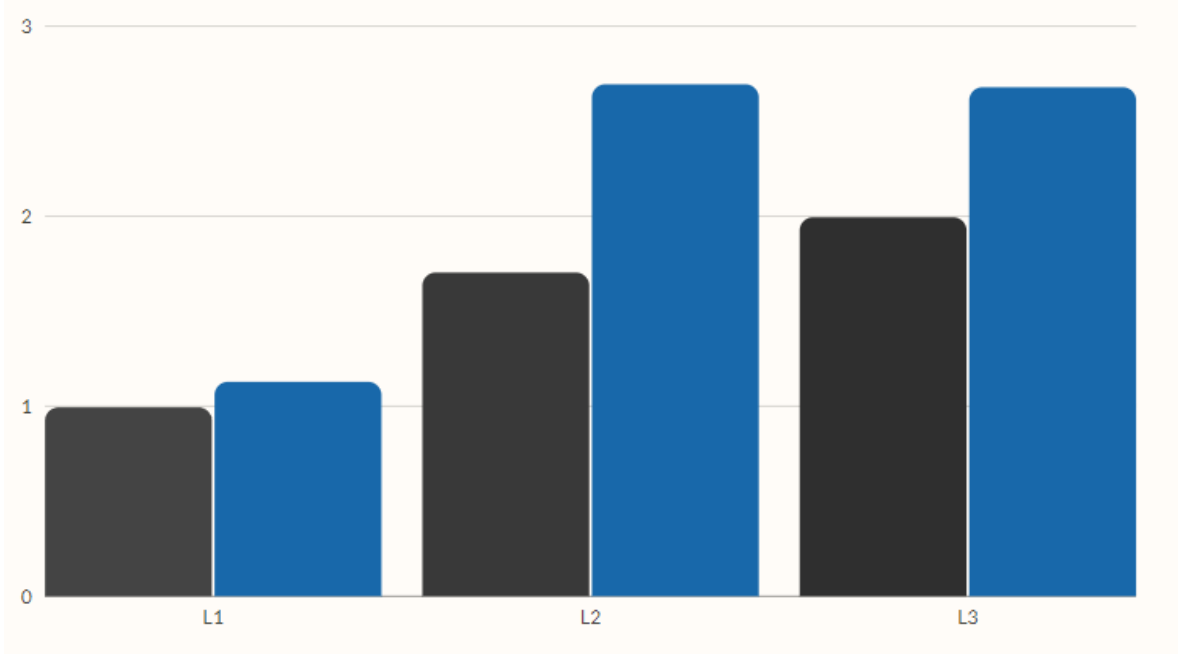
```
float example ()  
{  
    int i;  
  
    for (i=0;i<N_ELEM;i++)  
    {  
        a[i]=b[i]+8;  
    }  
  
    return a[N_ELEM-1];  
}
```



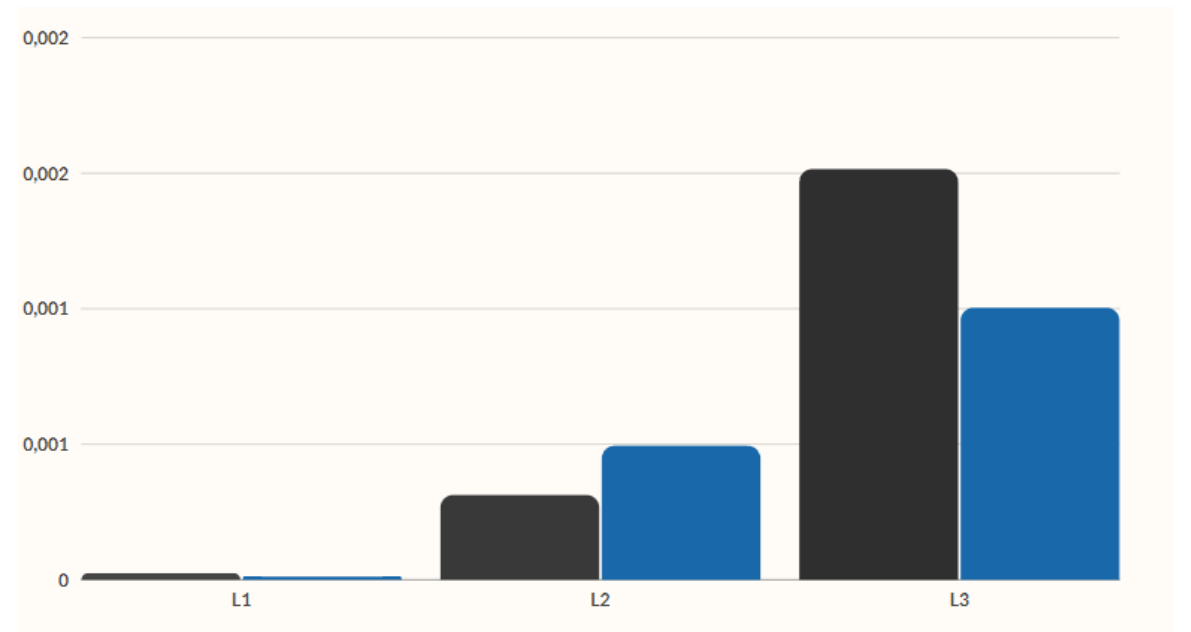
Equipo	Tipo de optimización	Tipo de instrucciones	Ciclos por elemento	Tiempo mínimo (segundos)
PC I (más potente)	/O2 (full optimization)	Instrucciones /SSE	0.90332	6.60714e-07
		Instrucciones /IA32	0.931641	6.81429e-07
	/Od (no optimization)	Instrucciones /SSE	4.86426	3.55786e-06
		Instrucciones /IA32	4.83594	3.53714e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 6x32KB -> 1024 * 60	0.990853	2.17421e-05
		L2 6x256KB -> 1024 * 500	1.70118	0.000311072
		L3 12*10^6B -> 1024 * 4096	1.99127	0.00151288
PC II (menos potente)	/O2 (full optimization)	Instrucciones /SSE	0.998047	7.3e-07
		Instrucciones /IA32	1.13965	8.33571e-07
	/Od (no optimization)	Instrucciones /SSE	5.2207	3.81857e-06
		Instrucciones /IA32	5.17578	3.78571e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 2x32KB -> 1024 * 25	1.12469	1.02829e-05
		L2 2x256KB -> 1024 * 500	2.69047	0.000491972
		L3 4*10^6B -> 1024 * 1024	2.67498	0.00100176

Diferencias dependiendo de la caché (Con /SSE y /O2).

Ciclos por elemento.



Tiempo mínimo (en segundos)



Bucles – Suma por iteración

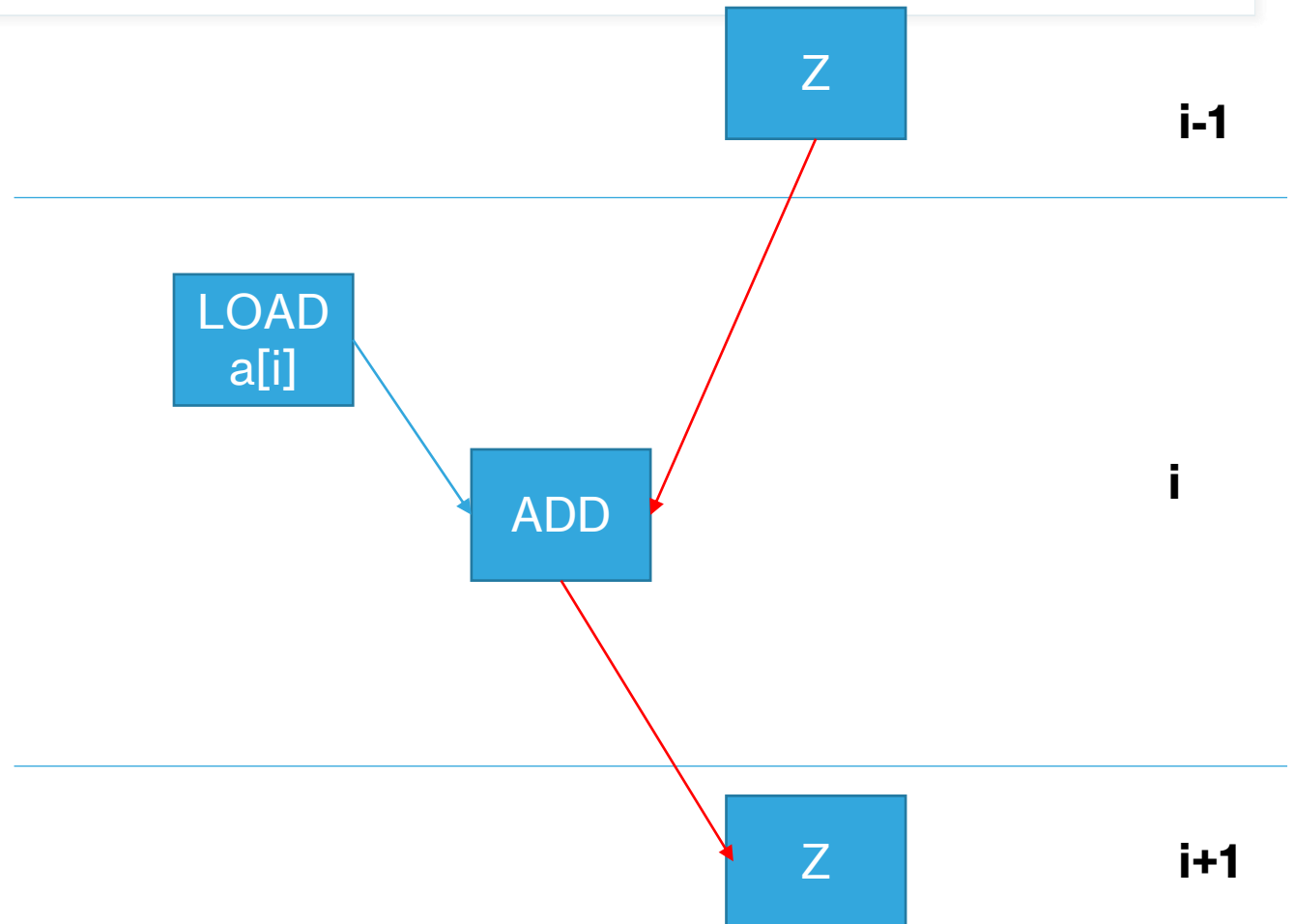
```
float problemA ()
{
    int i;
    float z = 0.0;

    for (i=0;i<N_ELEM;i++)
    {
        z = z + a[i];
    }

    return z;
}
```

Bucles – Suma por iteración

```
float problemA ()  
{  
    int i;  
    float z = 0.0;  
  
    for (i=0; i<N_ELEM; i++)  
    {  
        z = z + a[i];  
    }  
  
    return z;  
}
```



Equipo	Tipo de optimización	Tipo de instrucciones	Ciclos por elemento	Tiempo mínimo (segundos)
PC I (más potente)	/O2 (full optimization)	Instrucciones /SSE	3.24414	2.96929e-06
		Instrucciones /IA32	2.3584	1.725e-06
	/Od (no optimization)	Instrucciones /SSE	8.6543	6.33e-06
		Instrucciones /IA32	7.10254	5.195e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 6x32KB -> 1024 * 60	2.96904	6.51493e-05
		L2 6x256KB -> 1024 * 500	2.95112	0.000539634
		L3 12*10^6B -> 1024 * 4096	3.02621	0.00453316
PC II (menos potente)	/O2 (full optimization)	Instrucciones /SSE	3.45996	2.53071e-06
		Instrucciones /IA32	2.59375	1.89714e-06
	/Od (no optimization)	Instrucciones /SSE	7.79199	5.69929e-06
		Instrucciones /IA32	7.80957	5.71214e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 2x32KB -> 1024 * 25	3.35406	3.06657e-05
		L2 2x256KB -> 1024 * 500	4.134	0.000755932
		L3 4*10^6B -> 1024 * 1024	3.70883	0.00138893

Código ensamblador con /Od (y /SSE)

▶ `z = z + a[i];`

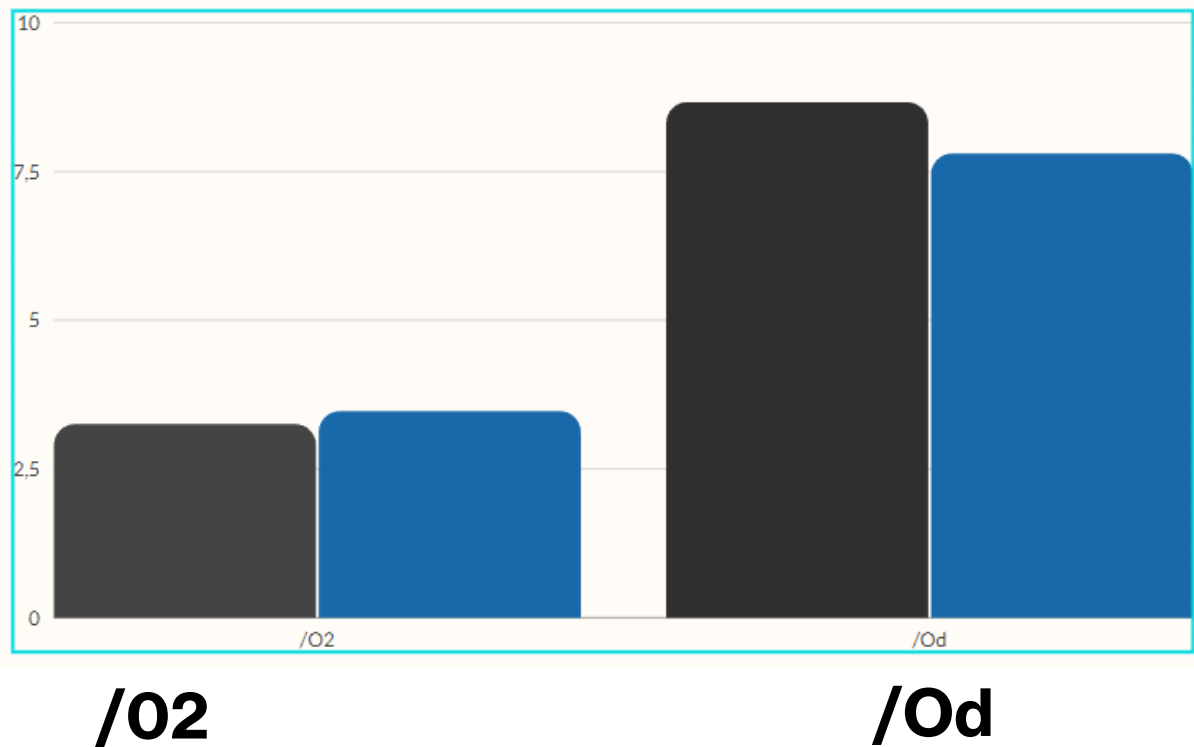
```
000F11E9    mov     ecx,dword ptr [i]
000F11EC    movss   xmm0,dword ptr [z]
000F11F1    addss   xmm0,dword ptr a (0FE3C0h)[ecx*4]
000F11FA    movss   dword ptr [z],xmm0
    }
```

Código ensamblador con /O2 (y /SSE)

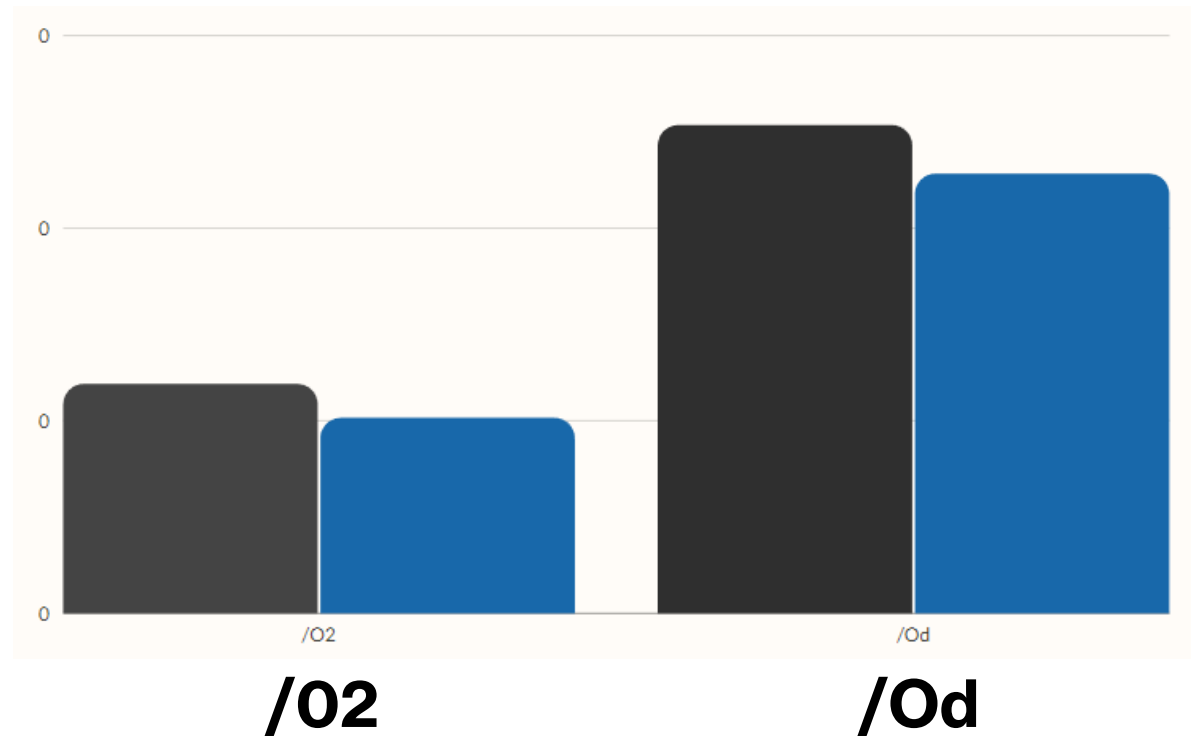
```
        z = z + a[i];  
00C01150  addss      xmm0,dword ptr [eax-4]  
00C01155  addss      xmm0,dword ptr [eax]  
00C01159  addss      xmm0,dword ptr [eax+4]  
00C0115E  addss      xmm0,dword ptr [eax+8]  
00C01163  addss      xmm0,dword ptr [eax+0Ch]  
00C01168  addss      xmm0,dword ptr [eax+10h]  
00C0116D  addss      xmm0,dword ptr [eax+14h]  
00C01172  addss      xmm0,dword ptr [eax+18h]  
00C01177  add        eax,20h  
00C0117A  cmp        eax,0C0F3C4h  
00C0117F  jl         problemA+10h (0C01150h)  
00C01181  movss      dword ptr [ebp-4],xmm0
```

Diferencias entre /O2 y /Od (Usando /SSE)

Ciclos por elemento.



Tiempo mínimo (en segundos)



Bucles – Dos sumas por iteración

```
float problemB ()
{
    int i;
    float z = 0.0;

    for (i=0;i<N_ELEM;i++)
    {
        // Suma de dos vectores
        z = z + (a[i] + b[i]);

        // Otra forma: Suma de un vector
        // y una constante
        // z = z + (a[i] + 3.0);
    }

    return z;
}
```

```
007811A0 movss xmm1,dword ptr b (07853C0h)[eax]
007811A8 addss xmm1,dword ptr a (078D3C0h)[eax]
007811B0 addss xmm1,xmm0
007811B4 movss xmm0,dword ptr [eax+78D3C4h]
007811BC addss xmm0,dword ptr [eax+7853C4h]
007811C4 addss xmm1,xmm0
007811C8 movss xmm0,dword ptr [eax+78D3C8h]
007811D0 addss xmm0,dword ptr [eax+7853C8h]
007811D8 addss xmm1,xmm0
007811DC movss xmm0,dword ptr [eax+78D3CCh]
007811E4 addss xmm0,dword ptr [eax+7853CCh]
007811EC addss xmm1,xmm0
007811F0 movss xmm0,dword ptr [eax+78D3D0h]
007811F8 addss xmm0,dword ptr [eax+7853D0h]
00781200 addss xmm1,xmm0
00781204 movss xmm0,dword ptr [eax+78D3D4h]
0078120C addss xmm0,dword ptr [eax+7853D4h]
00781214 addss xmm1,xmm0
00781218 movss xmm0,dword ptr [eax+78D3D8h]
00781220 addss xmm0,dword ptr [eax+7853D8h]
00781228 addss xmm1,xmm0
0078122C movss xmm0,dword ptr [eax+78D3DCh]
00781234 addss xmm0,dword ptr [eax+7853DCh]
0078123C add eax,20h
0078123F addss xmm0,xmm1
00781243 cmp eax,2000h
00781248 jl problemB+10h (07811A0h)
0078124E movss dword ptr [ebp-4],xmm0
```

Con /O2 y /SSE

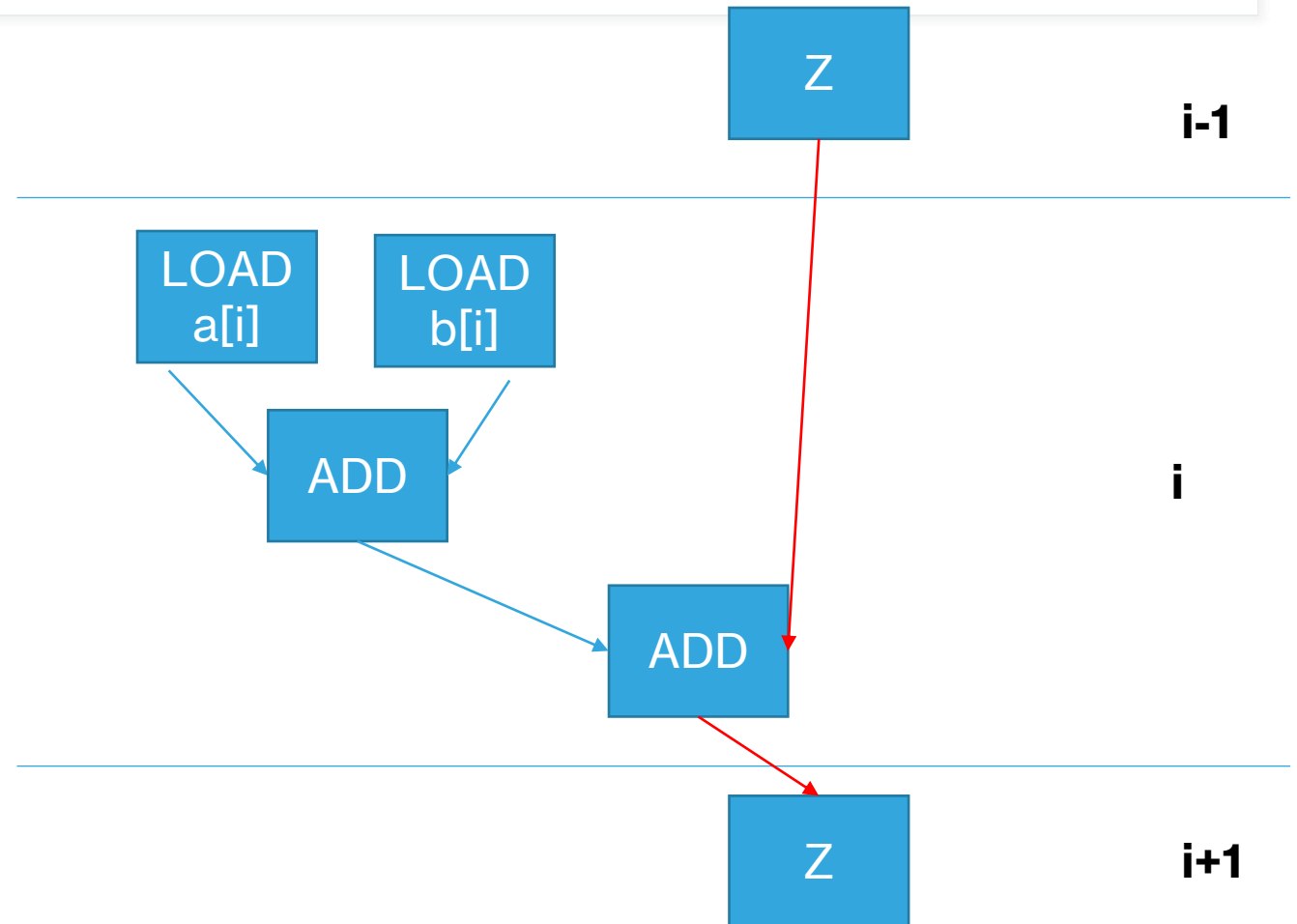
Bucles – Dos sumas por iteración

```
float problemB ()
{
    int i;
    float z = 0.0;

    for (i=0;i<N_ELEM;i++)
    {
        // Suma de dos vectores
        z = z + (a[i] + b[i]);

        // Otra forma: Suma de un vector
        // y una constante
        // z = z + (a[i] + 3.0);
    }

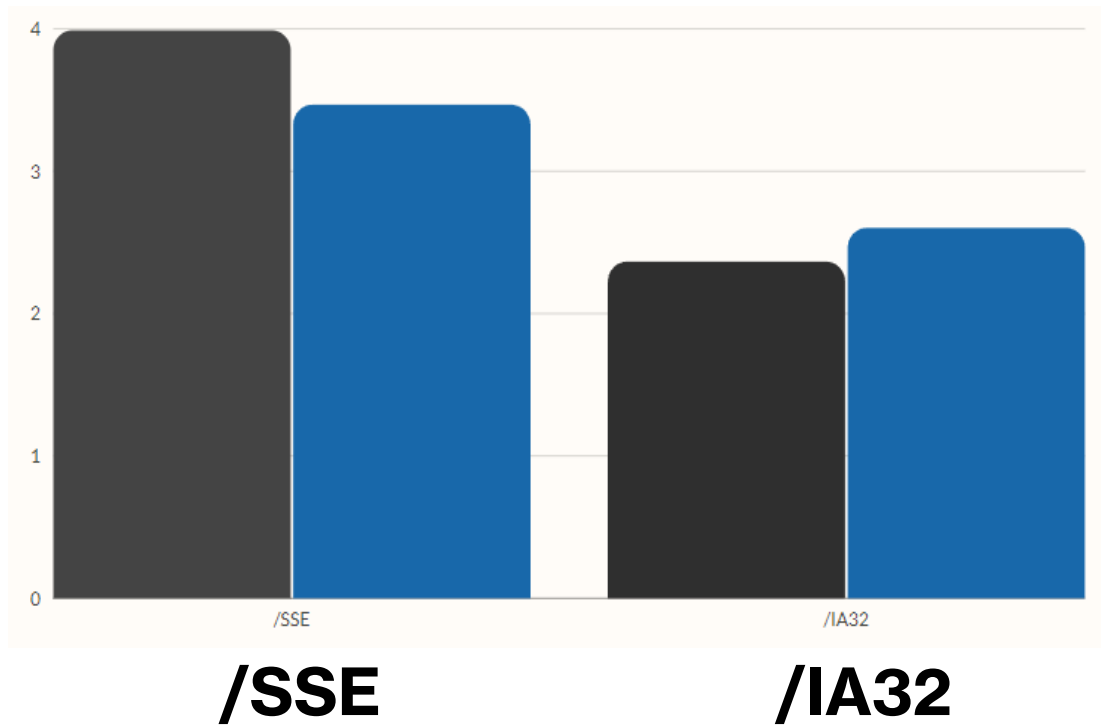
    return z;
}
```



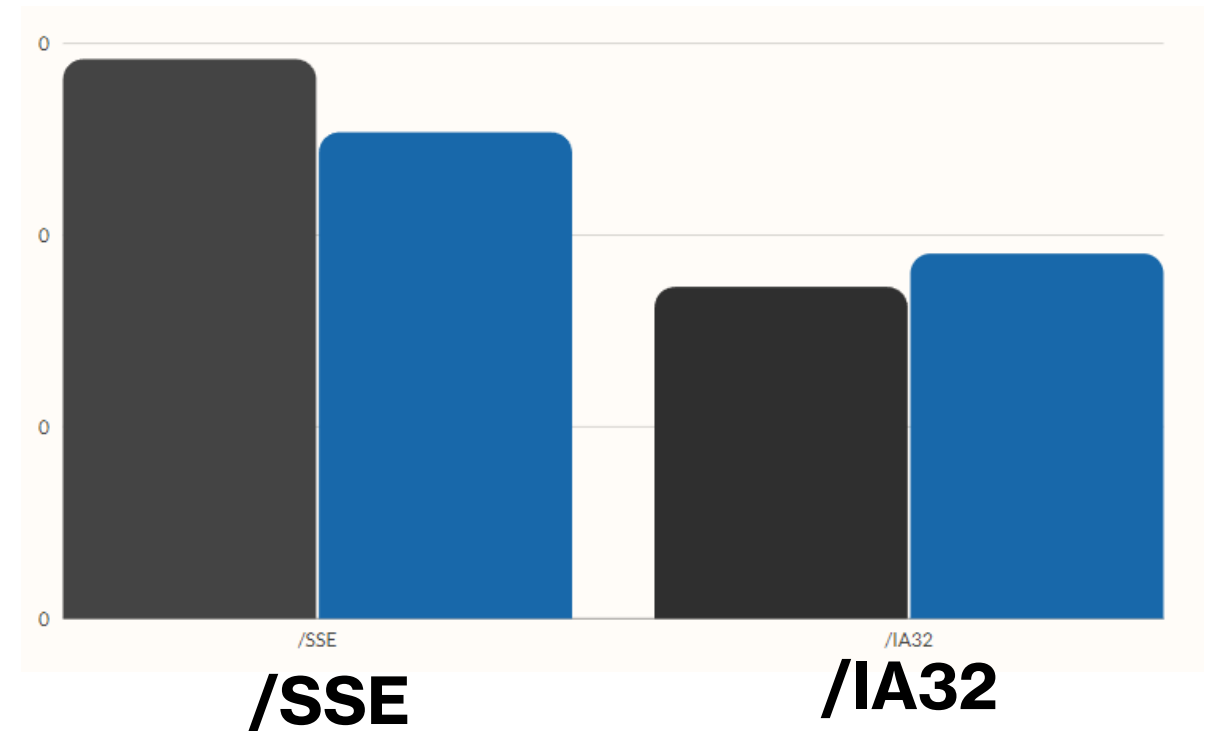
Equipo	Tipo de optimización	Tipo de instrucciones	Ciclos por elemento	Tiempo mínimo (segundos)
PC I (más potente)	/O2 (full optimization)	Instrucciones /SSE	3.98438	2.91429e-06
		Instrucciones /IA32	2.36035	1.72643e-06
	/Od (no optimization)	Instrucciones /SSE	7.89648	5.77571e-06
		Instrucciones /IA32	7.10254	4.89357e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 6x32KB -> 1024 * 60	2.96943	6.51579e-05
		L2 6x256KB -> 1024 * 500	3.15949	0.000577736
		L3 12*10^6B -> 1024 * 4096	3.27762	0.00490977
PC II (menos potente)	/O2 (full optimization)	Instrucciones /SSE	3.46289	2.53286e-06
		Instrucciones /IA32	2.59766	1.9e-06
	/Od (no optimization)	Instrucciones /SSE	7.79102	5.69857e-06
		Instrucciones /IA32	7.80664	5.71e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 2x32KB -> 1024 * 25	3.35336	3.06593e-05
		L2 2x256KB -> 1024 * 500	3.37362	0.000616891
		L3 4*10^6B -> 1024 * 1024	3.38093	0.00126613

Diferencias entre /SSE e /IA32 (Usando /O2)

Ciclos por elemento.



Tiempo mínimo.



Bucles – Suma condicional (muy fácilmente predecible por la BTB)

```
float problemD1 ()
```

```
{
```

```
    int i;
```

```
    float z = 0.0;
```

```
    for (i=0;i<N_ELEM;i++)
```

```
    {
```

```
        // Para valores MUY facilmente
```

```
        // predecibles (todos son 1)
```

```
        if (cond1[i] == 1) {
```

```
            z = z + a[i];
```

```
        }
```

```
        else {
```

```
            z = z + b[i];
```

```
        }
```

```
    }
```

```
    return z;
```

```
}
```

cond1[i]=1;

00B712B0	cmp	dword ptr cond1 (0B873C0h) [ecx], 1
00B712B7	mov	eax, offset b (0B753C0h)
00B712BC	cmov	eax, edx
00B712BF	addss	xmm0, dword ptr [eax+ecx]
00B712C4	add	ecx, 4
00B712C7	movss	dword ptr [ebp-4], xmm0
00B712CC	cmp	ecx, 2000h
00B712D2	jnl	problemD1+10h (0B712B0h)
00B712D4	fild	dword ptr [ebp-4]

Con /O2 y /SSE

Equipo	Tipo de optimización	Tipo de instrucciones	Ciclos por elemento	Tiempo mínimo (segundos)
PC I (más potente)	/O2 (full optimization)	Instrucciones /SSE	2.97852	2.17857e-06
		Instrucciones /IA32	2.43359	1.78e-06
	/Od (no optimization)	Instrucciones /SSE	8.9209	6.525e-06
		Instrucciones /IA32	6.68652	4.89071e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 6x32KB -> 1024 * 60	2.91781	6.4025e-05
		L2 6x256KB -> 1024 * 500	3.28757	0.000601156
		L3 12*10^6B -> 1024 * 4096	3.40077	0.00509423
PC II (menos potente)	/O2 (full optimization)	Instrucciones /SSE	3.39551	2.48357e-06
		Instrucciones /IA32	2.59473	1.8978e-06
	/Od (no optimization)	Instrucciones /SSE	7.78809	5.69643e-06
		Instrucciones /IA32	7.79833	5.70429e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 2x32KB -> 1024 * 25	3.36016	3.07214e-05
		L2 2x256KB -> 1024 * 500	3.87059	0.000707766
		L3 4*10^6B -> 1024 * 1024	3.90005	0.00146053

Código ensamblador /SSE, /Od de la Suma Condicional

```
    for (i=0;i<N_ELEM;i++)
007012EE  mov     dword ptr [i],0
007012F5  jmp     problemD1+20h (0701300h)
007012F7  mov     eax,dword ptr [i]
007012FA  add     eax,1
007012FD  mov     dword ptr [i],eax
00701300  cmp     dword ptr [i],800h
00701307  jge     problemD1+66h (0701346h)
    {
        // Para valores MUY facilmente predecibles (todos son 1)
        if (cond1[i] == 1) {
00701309  mov     ecx,dword ptr [i]
0070130C  cmp     dword ptr cond1 (07183C0h)[ecx*4],1
00701314  jne     problemD1+4Eh (070132Eh) ▶
```

```
        z = z + a[i];
00701316  mov     edx,dword ptr [i]
00701319  movss   xmm0,dword ptr [z]
0070131E  addss   xmm0,dword ptr a (070E3C0h)[edx*4]
00701327  movss   dword ptr [z],xmm0
    }
0070132C  jmp     problemD1+64h (0701344h)
    else {
        z = z + b[i];
0070132E  mov     eax,dword ptr [i]
00701331  movss   xmm0,dword ptr [z]
00701336  addss   xmm0,dword ptr b (07063C0h)[eax*4]
0070133F  movss   dword ptr [z],xmm0
    }
```

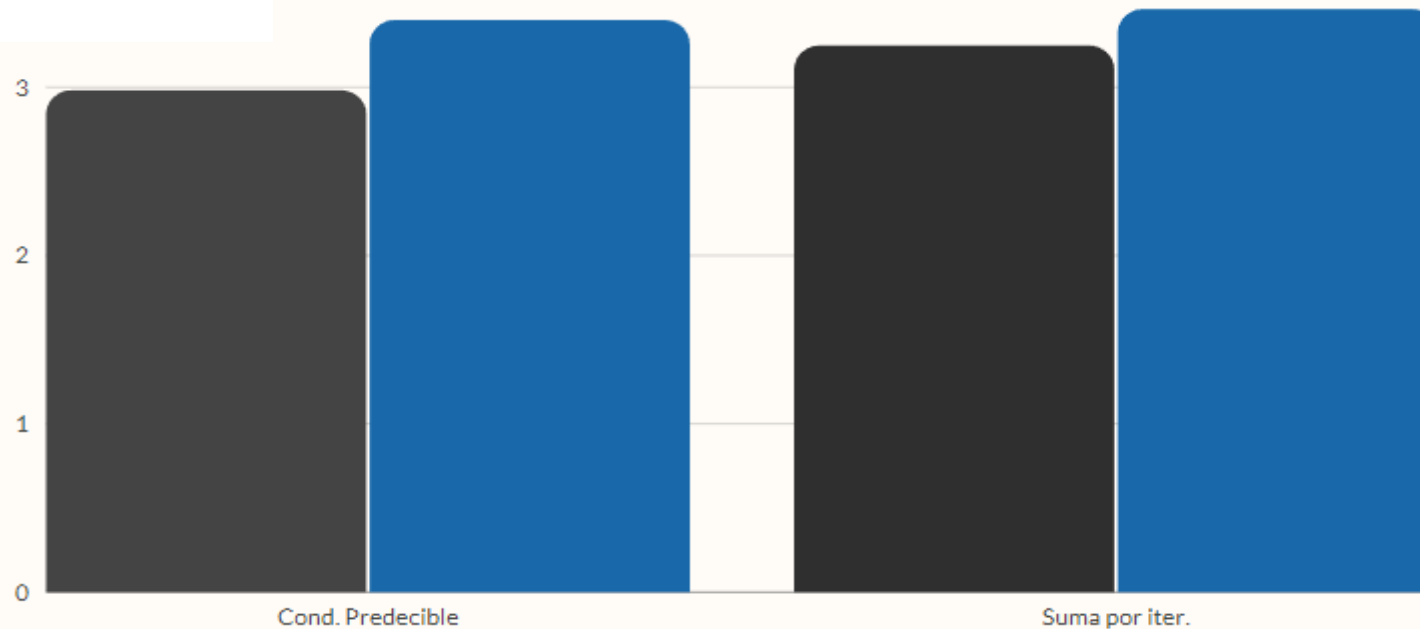
Diferencias de ciclos por elemento entre condición muy fácilmente predecible y suma por iteración

Condición fácilmente predecible

Suma por iteración (sin cond.)

```
// Para valores MUY facilmente  
// predecibles (todos son 1)  
if (cond1[i] == 1) {  
    z = z + a[i];  
}  
else {  
    z = z + b[i];  
}
```

```
for (i=0;i<N_ELEM;i++)  
{  
    z = z + a[i];  
}
```



Bucles – Suma condicional (fácilmente predecible por la BTB)

```
float problemD2()
```

```
{
```

```
    int i;
```

```
    float z = 0.0;
```

```
    for (i = 0; i < N_ELEM; i++)
```

```
    {
```

```
        // Para valores predecibles
```

```
        // (entre 0 y 1)
```

```
        if (cond2[i] == 1) {
```

```
            z = z + a[i];
```

```
        }
```

```
        else {
```

```
            z = z + b[i];
```

```
        }
```

```
    }
```

```
    return z;
```

```
}
```

```
cond2[i+0]=1;
```

```
cond2[i+1]=0;
```

```
cond2[i+2]=1;
```

```
cond2[i+3]=1;
```

```
cond2[i+4]=0;
```

```
cond2[i+5]=1;
```

```
cond2[i+6]=1;
```

```
cond2[i+7]=1;
```

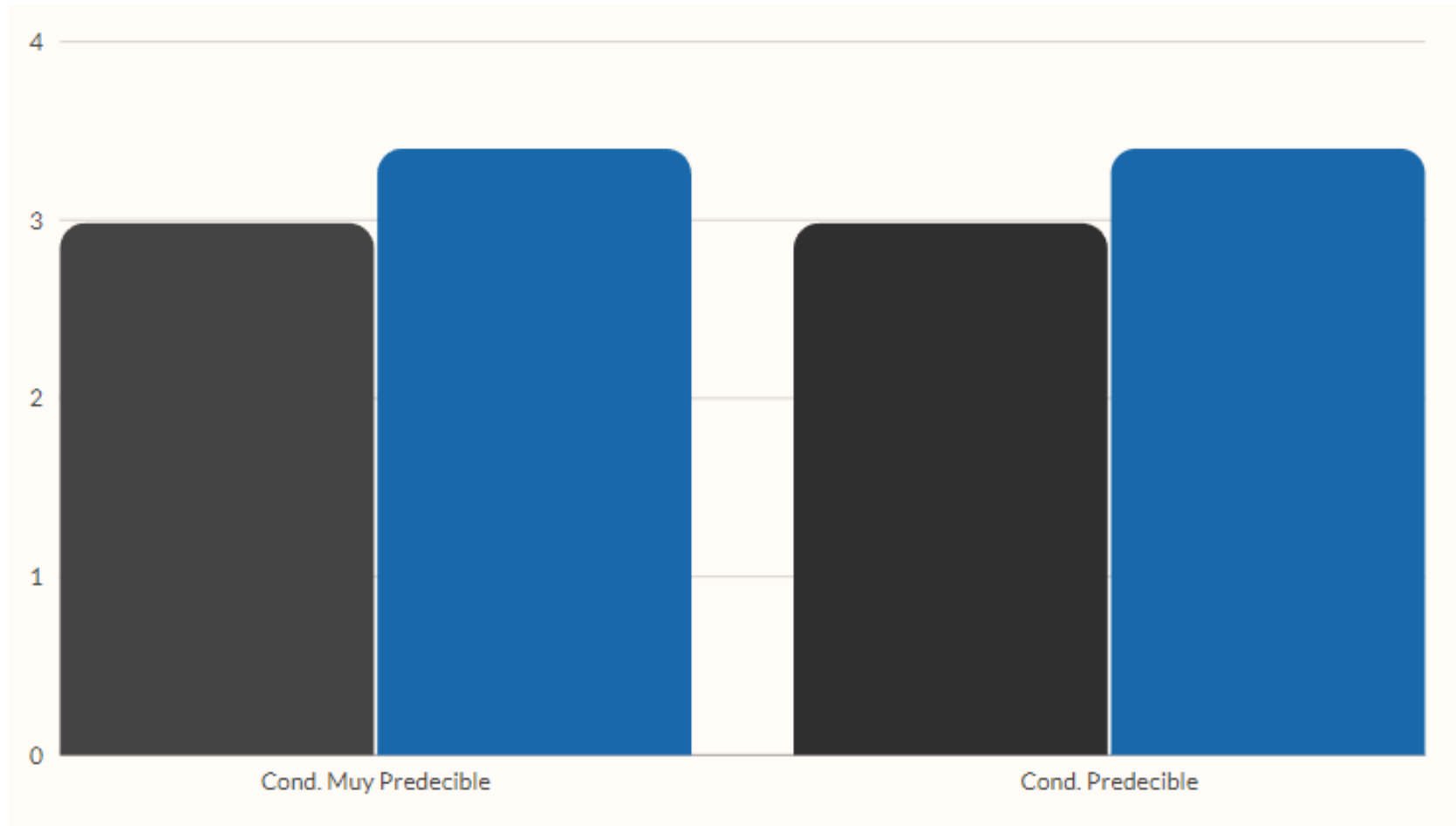
```
009612F0    cmp     dword ptr cond2 (09753C0h) [ecx],1
009612F7    mov     eax,offset b (09653C0h)
009612FC    cmov     eax,edx
009612FF    addss    xmm0,dword ptr [eax+ecx]
00961304    add     ecx,4
00961307    movss    dword ptr [ebp-4],xmm0
0096130C    cmp     ecx,2000h
00961312    jl      problemD2+10h (09612F0h)
00961314    fld     dword ptr [ebp-4]
```

```
dword ptr cond2 (09753C0h) [ecx],1
eax,offset b (09653C0h)
eax,edx
xmm0,dword ptr [eax+ecx]
ecx,4
dword ptr [ebp-4],xmm0
ecx,2000h
problemD2+10h (09612F0h)
dword ptr [ebp-4]
```

Con /O2 y /SSE

Equipo	Tipo de optimización	Tipo de instrucciones	Ciclos por elemento	Tiempo mínimo (segundos)
PC I (más potente)	/O2 (full optimization)	Instrucciones /SSE	2.97754	2.17786e-06
		Instrucciones /IA32	2.6582	1.94429e-06
	/Od (no optimization)	Instrucciones /SSE	9.03809	6.61071e-06
		Instrucciones /IA32	7.10742	5.19857e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 6x32KB -> 1024 * 60	2.90999	6.38536e-05
		L2 6x256KB -> 1024 * 500	3.29595	0.000602689
		L3 12*10^6B -> 1024 * 4096	3.5364	3.5364
PC II (menos potente)	/O2 (full optimization)	Instrucciones /SSE	3.3916	2.48071e-06
		Instrucciones /IA32	2.8252	2.06643e-06
	/Od (no optimization)	Instrucciones /SSE	7.79395	5.70071e-06
		Instrucciones /IA32	7.79102	5.69857e-06
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 2x32KB -> 1024 * 25	3.36109	3.073e-05
		L2 2x256KB -> 1024 * 500	4.06545	0.000743396
		L3 4*10^6B -> 1024 * 1024	4.18752	0.00156819

Comparación de ciclos por elemento entre condición muy fácilmente predecible y condición predecible.



Bucles – Suma condicional (difícilmente predecible por la BTB)

```
float problemC ()
```

```
{
```

```
    int    i;
```

```
    float  z = 0.0;
```

```
    for (i=0;i<N_ELEM;i++)
```

```
    {
```

```
        // cond3 contiene valores aleatorios
```

```
        // (entre 0 y 1) difícilmente predecibles
```

```
        if (cond3[i] == 1) {
```

```
            z = z + a[i];
```

```
        }
```

```
        else {
```

```
            z = z + b[i];
```


```
        }
```

```
    }
```

```
    return z;
```

```
}
```

cond3[i]=rand()%2;



00C91270	cmp	dword ptr cond3 (0C9F3C0h) [ecx], 1
00C91277	mov	eax, offset b (0C953C0h)
00C9127C	cmov	eax, edx
00C9127F	addss	xmm0, dword ptr [eax+ecx]
00C91284	add	ecx, 4
00C91287	movss	dword ptr [ebp-4], xmm0
00C9128C	cmp	ecx, 2000h
00C91292	jnl	problemC+10h (0C91270h)
00C91294	fild	dword ptr [ebp-4]

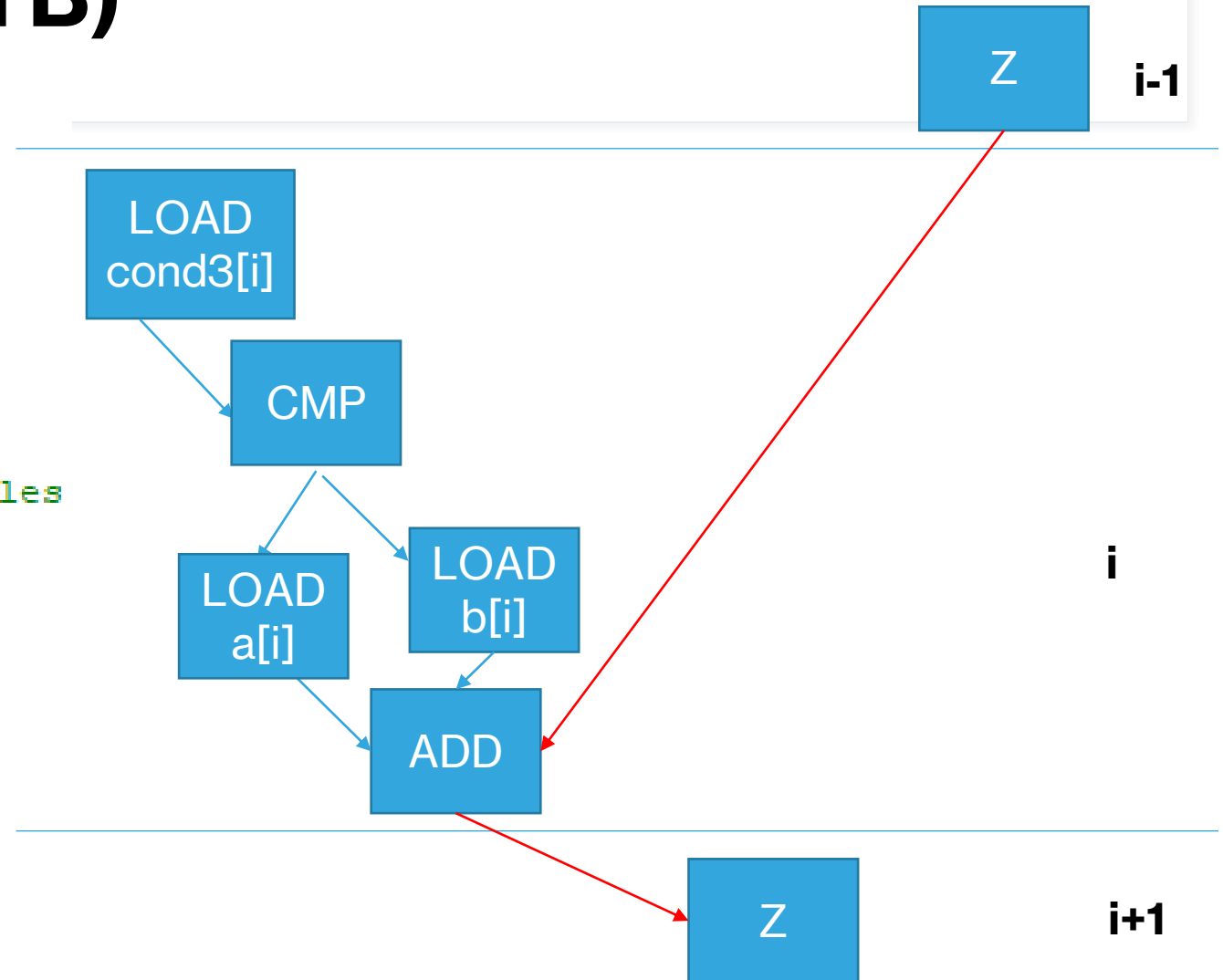
Con /O2 y /SSE

Bucles – Suma condicional (difícilmente predecible por la BTB)

```
float problemC ()
{
    int i;
    float z = 0.0;

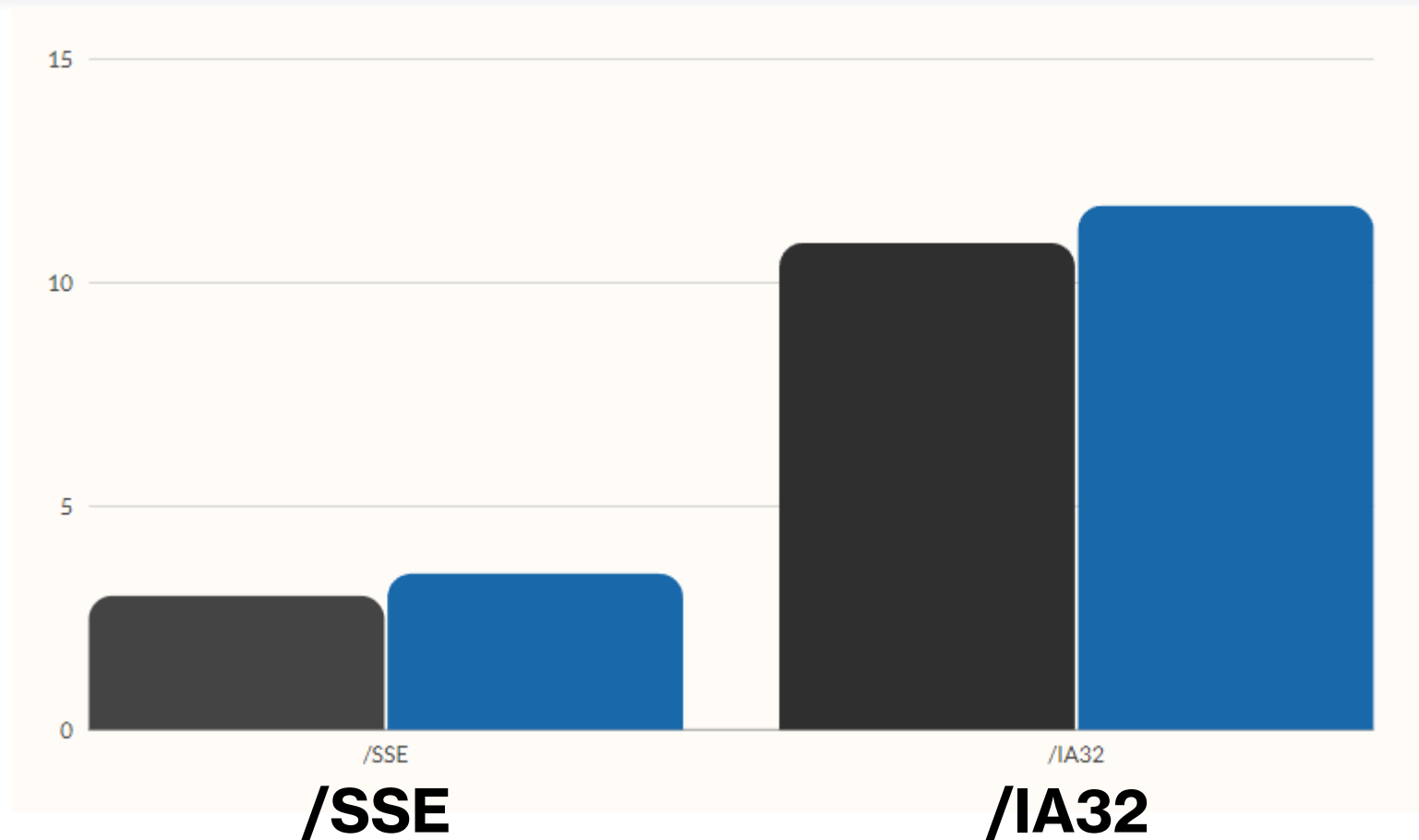
    for (i=0;i<N_ELEM;i++)
    {
        // cond3 contiene valores aleatorios
        // (entre 0 y 1) difícilmente predecibles
        if (cond3[i] == 1) {
            z = z + a[i];
        }
        else {
            z = z + b[i];
        }
    }

    return z;
}
```



Equipo	Tipo de optimización	Tipo de instrucciones	Ciclos por elemento	Tiempo mínimo (segundos)
PC I (más potente)	/O2 (full optimization)	Instrucciones /SSE	2.97949	2.17929e-06
		Instrucciones /IA32	10.874	7.95357e-06
	/Od (no optimization)	Instrucciones /SSE	18.2354	1.33379e-05
		Instrucciones /IA32	16.9834	1.24221e-05
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 6x32KB -> 1024 * 60	2.91406	6.39429e-05
		L2 6x256KB -> 1024 * 500	3.34541	0.000611733
		L3 12*10^6B -> 1024 * 4096	3.52593	0.00528173
PC II (menos potente)	/O2 (full optimization)	Instrucciones /SSE	3.47266	2.54e-06
		Instrucciones /IA32	11.7041	8.56071e-06
	/Od (no optimization)	Instrucciones /SSE	19.4238	1.42071e-05
		Instrucciones /IA32	19.4297	1.42114e-05
	Vectores grandes (para causar fallos en caché) con /O2 y /SSE	L1 2x32KB -> 1024 * 25	3.36133	3.07321e-05
		L2 2x256KB -> 1024 * 500	4.24129	0.000775549
		L3 4*10^6B -> 1024 * 1024	4.23225	0.00158494

Resultados ciclos por elemento – Suma condicional (difícilmente predecible por la BTB)





**¿Alguna
pregunta?**

