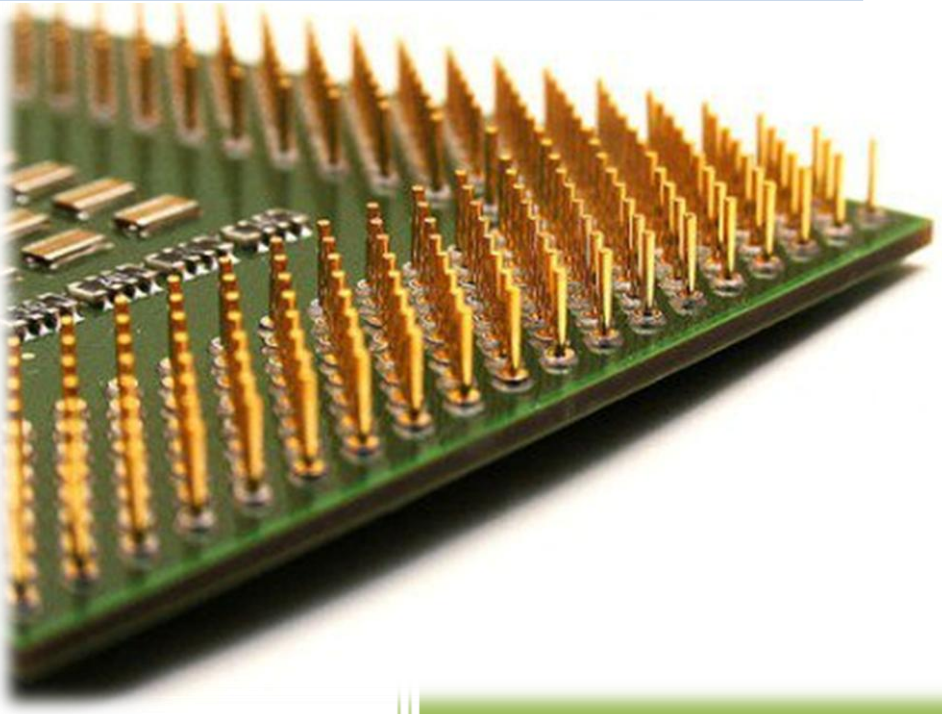
2012

Memoria práctica de Ingeniería de Computadores II: Programación en el ensamblador del procesador segmentado DLX del bucle DAXPY



Longinos Recuero Bustos
(lrecuero1@alumno.uned.es)
longinox.blogspot.com
675727441
Alcázar de San Juan
19/02/2012

Tabla de contenido

Apartado a 3
Apartado b 3
Apartado c 4
Apartado d 5
Apartado e 6
Apéndice A 7
Apéndice B 8
Apéndice C 9
Apéndice D 10

Apartado a

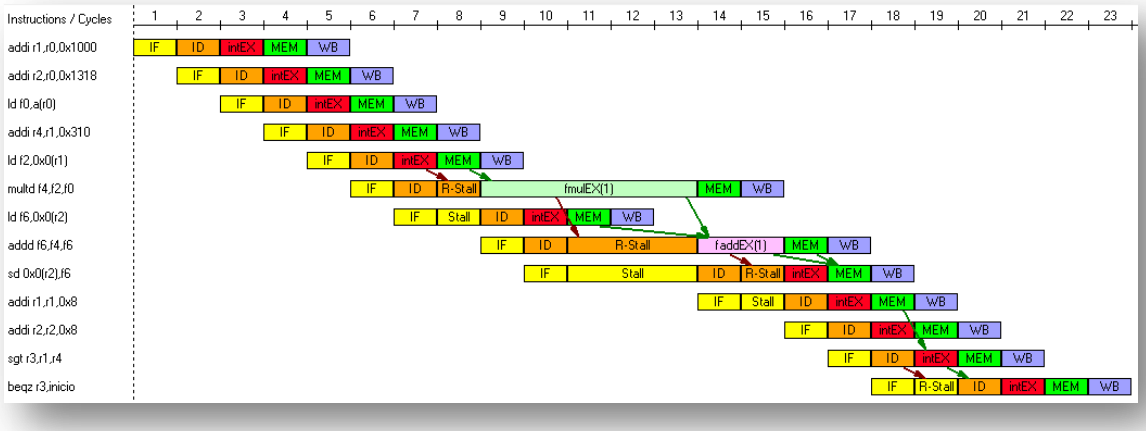
La programación de este apartado se encuentra en el listado del Apéndice A y en el fichero `BUCLE.S` entregado junto a la práctica.

Apartado b

En la siguiente tabla se resumen algunos de los valores necesarios para la resolución de este apartado:

	DETENCIONES						CICLO		
INSTRUCCIÓN	**Nº**	**IF**	**ID**	**EX**	**MEM**	**WB**	**INICIO**	**FINAL**	**TOTAL**
add r1,r0,x	0	---	---	---	---	---	1	5	5
add r2,r0,y	0	---	---	---	---	---	2	6	5
ld f0,a	0	---	---	---	---	---	3	7	5
addi r4,r1, 784	0	---	---	---	---	---	4	8	5
ld f2, 0(r1)	0	---	---	---	---	---	5	9	**5**
multd f4, f2, f0	1	---	RAW	---	---	---	6	15	**10**
ld f6, 0(r2)	1	STRUCTURAL	---	---	---	---	7	12	**6**
addd f6,f4,f6	3	RAW	---	---	---	---	9	17	**9**
sd 0(r2), f6	3 + 1	STRUCTURAL	RAW	---	---	---	10	18	**9**
addi r1, r1, 8	1	STRUCTURAL	---	---	---	---	14	19	**6**
addi r2,r2, 8	0	---	---	---	---	---	16	20	**5**
sgt r3, r1, r4	0	---	---	---	---	---	17	21	**5**
beqz r3, inicio	1	RAW	---	---	---	---	18	23	**6**

Todo lo anterior y algunos datos más son recogidos de manera gráfica en la siguiente imagen capturada del simulador WinDLX:



En resumen:

- Hay un total de 11 detenciones.
- En el ciclo 5, comienza la ejecución de la instrucción en la primera iteración del bucle.
- Cada iteración del bucle original consume 61 ciclos.

Apartado c

La programación de este apartado se encuentra en el listado del Apéndice B y en el fichero `BUCLE_C.S` entregado junto a la práctica.

Como tamaño del problema, se ha elegido usar dos vectores de 99 elementos cada uno.

La siguiente tabla, resume parte de las estadísticas generadas por WinDLX para ambos códigos:

	BUCLE.S	BUCLE_C.S
CICLOS DE RELOJ DEL PROGRAMA	1592	1196
NUMERO DE INSTRUCCIONES	896	632

En ella se aprecia como el nuevo código desenrollado mejora significativamente al código original.

Esto es así porque el desenrollamiento ha permitido que el número de iteraciones necesarias para recorrer el bucle se haya dividido por tres y el total de instrucciones ejecutadas sea inferior ya que se han eliminado 8 instrucciones por cada iteración del bucle desenrollado:

$$\frac{99\ iteraciones}{3} \times 8\ instrucciones = 264\ instrucciones\ eliminadas$$

Coincidiendo con lo recogido anteriormente:

$$896 - 264 = 632\ instrucciones\ ejecutadas$$

La aceleración (speedUp), al introducir una mejora, viene dada por la siguiente ecuación:

$$Sp = \frac{Tiempo_{CPU_{Original}}}{Tiempo_{CPU_{mejorada}}} \equiv \frac{Num_{Inst_{Orig}} \times CPI_{Orig} \times Tiempo_{Ciclo}}{Num_{Inst_{Mejorada}} \times CPI_{Mejorada} \times Tiempo_{Ciclo}}$$

Puesto que ambos programas se ejecutan sobre el mismo procesador, la ecuación fina queda como:

$$Sp = \frac{Ciclos\_reloj_{programa\_origen}}{Ciclos\_reloj_{programa\_mejorado}}$$

Por lo tanto, la aceleración obtenida sobre el bucle original es de:

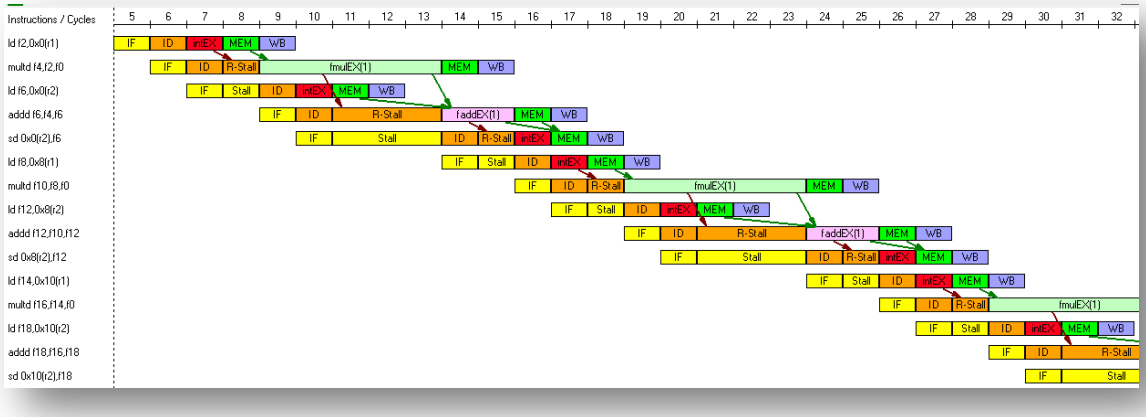
$$\frac{1592\ ciclos}{1196\ ciclos} = 1{,}33$$

Apartado d

Como primera aproximación, se puede planificar el código del apartado c, reorganizando (agrupando) las instrucciones por tipo (véase el listado de código del Apéndice C y en el fichero `BUCLE_D.S`). Sin embargo la ganancia obtenida, apenas mejora lo obtenido con anterioridad.

$$\frac{1592\ ciclos}{1064\ ciclos} = 1{,}49$$

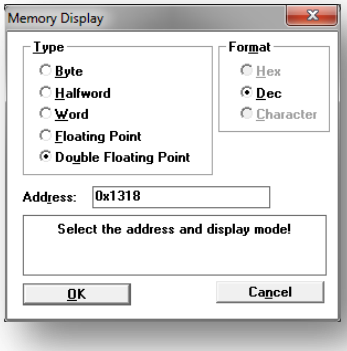
Por tanto, se propone la eliminación del máximo de dependencias que hubiese en el código del apartado c, reordenando el código significativamente. Las dependencias a eliminar se pueden observar en la siguiente imagen:

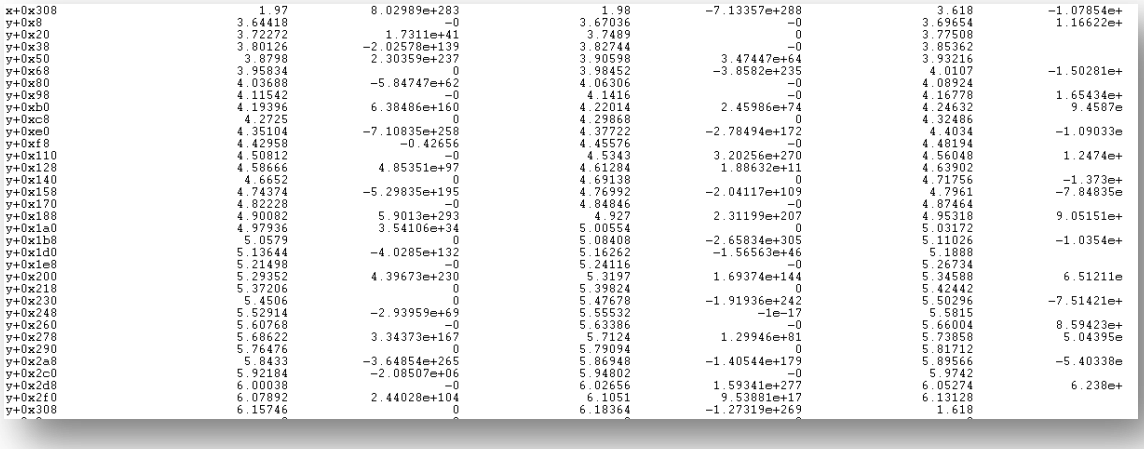


Esta reorganización da lugar a la creación del listado del Apéndice D y el fichero `BUCLE_D_OP2.S`. Ahora se observa como esta nueva planificación es mucho más eficiente, consiguiendo una aceleración de:

$$\frac{1592\ ciclos}{767\ ciclos} = 2{,}07$$

Es importante darse cuenta de que esta reorganización de código, se hace manteniendo la semántica original del programa, es decir, el orden de las lecturas y las escrituras de los registros y la memoria. Para demostrar esto se puede recurrir a ver el estado de la memoria una vez finalizado el programa y compararla con la del programa original:





x+0x308	1.97	8.02989e+283	1.98	-7.13357e+288	3.618	-1.07854e+
y+0x8	3.64418	-0	3.67036	-0	3.69654	1.16622e+
y+0x20	3.72272	1.7311e+41	3.7489	0	3.77508	
y+0x38	3.80126	-2.02578e+139	3.82744	-0	3.85362	
y+0x50	3.8798	2.30359e+237	3.90598	3.47447e+64	3.93216	
y+0x68	3.95834	0	3.98452	-3.8582e+235	4.0107	-1.50281e+
y+0x80	4.03688	-5.84747e+62	4.06306	-0	4.08924	
y+0x98	4.11542	-0	4.1416	-0	4.16778	1.65434e+
y+0xb0	4.19396	6.38486e+160	4.22014	2.45986e+74	4.24632	9.4587e
y+0xc8	4.2725	0	4.29868	0	4.32486	
y+0xe0	4.35104	-7.10835e+258	4.37722	-2.78494e+172	4.4034	-1.09033e
y+0xf8	4.42958	-0.42656	4.45576	-0	4.48194	
y+0x110	4.50812	-0	4.5343	3.20256e+270	4.56048	1.2474e+
y+0x128	4.58666	4.85351e+97	4.61284	1.88632e+11	4.63902	
y+0x140	4.6652	0	4.69138	0	4.71756	-1.373e+
y+0x158	4.74374	-5.29835e+195	4.76992	-2.04117e+109	4.7961	-7.84835e
y+0x170	4.82228	-0	4.84846	-0	4.87464	
y+0x188	4.90082	5.9013e+293	4.927	2.31199e+207	4.95318	9.05151e+
y+0x1a0	4.97936	3.54106e+34	5.00554	0	5.03172	
y+0x1b8	5.0579	0	5.08408	-2.65834e+305	5.11026	-1.0354e+
y+0x1d0	5.13644	-4.0285e+132	5.16262	-1.56563e+46	5.1888	
y+0x1e8	5.21498	-0	5.24116	-0	5.26734	
y+0x200	5.29352	4.39673e+230	5.3197	1.69374e+144	5.34588	6.51211e
y+0x218	5.37206	0	5.39824	0	5.42442	
y+0x230	5.4506	0	5.47678	-1.91936e+242	5.50296	-7.51421e+
y+0x248	5.52914	-2.93959e+69	5.55532	-1e-17	5.5815	
y+0x260	5.60768	-0	5.63386	-0	5.66004	8.59423e+
y+0x278	5.68622	3.34373e+167	5.7124	1.29946e+81	5.73858	5.04395e
y+0x290	5.76476	0	5.79094	0	5.81712	
y+0x2a8	5.8433	-3.64854e+265	5.86948	-1.40544e+179	5.89566	-5.40338e
y+0x2c0	5.92184	-2.08507e+06	5.94802	-0	5.9742	
y+0x2d8	6.00038	-0	6.02656	1.59341e+277	6.05274	6.238e+
y+0x2f0	6.07892	2.44028e+104	6.1051	9.53881e+17	6.13128	
y+0x308	6.15746	0	6.18364	-1.27319e+269	1.618	

Apartado e

El CPI medio viene descrito por la expresión:

$$\frac{Ciclos\ de\ reloj\ del\ programa}{Número\ de\ instrucciones}$$

Por tanto, la siguiente tabla, resume los CPI medios que se obtienen en cada iteración del bucle en los apartados (a), (c), (d) y otros datos relacionados que dan una idea global de todo el desarrollo anterior:

	CICLOS	INSTRUCCIONES	DETENCIONES	ACELERACIÓN	CPI medio
BUCLE.S (a)	1592	869	695	---	19 / 09 = **2,11**
BUCLE_C.S (c)	1196	632	563	1,33	39 / 19 = **2,05**
BUCLE_D.S	1064	632	398	1,49	35 / 19 = **1,84**
BUCLE_D_OP2.S (d)	767	632	68	2,07	26 / 19 = **1,36**

Apéndice A

Listado del fichero

```
        .data
        .align 2

x:      .double 1.00, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09
        .double 1.10, 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19
        .double 1.20, 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29
        .double 1.30, 1.31, 1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39
        .double 1.40, 1.41, 1.42, 1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49
        .double 1.50, 1.51, 1.52, 1.53, 1.54, 1.55, 1.56, 1.57, 1.58, 1.59
        .double 1.60, 1.61, 1.62, 1.63, 1.64, 1.65, 1.66, 1.67, 1.68, 1.69
        .double 1.70, 1.71, 1.72, 1.73, 1.74, 1.75, 1.76, 1.77, 1.78, 1.79
        .double 1.80, 1.81, 1.82, 1.83, 1.84, 1.85, 1.86, 1.87, 1.88, 1.89
        .double 1.90, 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97, 1.98

y:      .double 2.00, 2.01, 2.02, 2.03, 2.04, 2.05, 2.06, 2.07, 2.08, 2.09
        .double 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19
        .double 2.20, 2.21, 2.22, 2.23, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29
        .double 2.30, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37, 2.38, 2.39
        .double 2.40, 2.41, 2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49
        .double 2.50, 2.51, 2.52, 2.53, 2.54, 2.55, 2.56, 2.57, 2.58, 2.59
        .double 2.60, 2.61, 2.62, 2.63, 2.64, 2.65, 2.66, 2.67, 2.68, 2.69
        .double 2.70, 2.71, 2.72, 2.73, 2.74, 2.75, 2.76, 2.77, 2.78, 2.79
        .double 2.80, 2.81, 2.82, 2.83, 2.84, 2.85, 2.86, 2.87, 2.88, 2.89
        .double 2.90, 2.91, 2.92, 2.93, 2.94, 2.95, 2.96, 2.97, 2.98

a:      .double 1.6180

        .text
        .global main

main:
        add r1,r0,x           ; R1 <= Dirección de comienzo de x
        add r2,r0,y           ; R2 <= Dirección de comienzo de y
        ld f0,a               ; F0 <= Valor de a
        addi r4,r1, 784       ; R4 <= Última posición del vector x

inicio:
        ld f2, 0(r1)          ; carga X(i)
        multd f4, f2, f0      ; multiplica a * X(i)
        ld f6, 0(r2)          ; carga Y(i)
        addd f6,f4,f6         ; suma a * X(i) + Y(i)
        sd 0(r2), f6          ; almacena Y(i)
        addi r1, r1, 8        ; incrementa índice X
        addi r2,r2, 8         ; incremente índice Y
        sgt r3, r1, r4        ; test por si finalizado
        beqz r3, inicio       ; bucle si no finalizado
        trap 0                ; Fin
```

Apéndice B

```
        .data
        .align 2

x:      .double 1.00, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09
        .double 1.10, 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19
        .double 1.20, 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29
        .double 1.30, 1.31, 1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39
        .double 1.40, 1.41, 1.42, 1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49
        .double 1.50, 1.51, 1.52, 1.53, 1.54, 1.55, 1.56, 1.57, 1.58, 1.59
        .double 1.60, 1.61, 1.62, 1.63, 1.64, 1.65, 1.66, 1.67, 1.68, 1.69
        .double 1.70, 1.71, 1.72, 1.73, 1.74, 1.75, 1.76, 1.77, 1.78, 1.79
        .double 1.80, 1.81, 1.82, 1.83, 1.84, 1.85, 1.86, 1.87, 1.88, 1.89
        .double 1.90, 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97, 1.98

y:      .double 2.00, 2.01, 2.02, 2.03, 2.04, 2.05, 2.06, 2.07, 2.08, 2.09
        .double 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19
        .double 2.20, 2.21, 2.22, 2.23, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29
        .double 2.30, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37, 2.38, 2.39
        .double 2.40, 2.41, 2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49
        .double 2.50, 2.51, 2.52, 2.53, 2.54, 2.55, 2.56, 2.57, 2.58, 2.59
        .double 2.60, 2.61, 2.62, 2.63, 2.64, 2.65, 2.66, 2.67, 2.68, 2.69
        .double 2.70, 2.71, 2.72, 2.73, 2.74, 2.75, 2.76, 2.77, 2.78, 2.79
        .double 2.80, 2.81, 2.82, 2.83, 2.84, 2.85, 2.86, 2.87, 2.88, 2.89
        .double 2.90, 2.91, 2.92, 2.93, 2.94, 2.95, 2.96, 2.97, 2.98

a:      .double 1.6180

        .text
        .global main

main:
        add r1,r0,x           ; R1 <= Dirección de comienzo de x
        add r2,r0,y           ; R2 <= Dirección de comienzo de y
        ld f0,a               ; F0 <= Valor de a
        addi r4,r1, 784       ; R4 <= Última posición del vector x

iter1:
        ld f2, 0(r1)          ; carga X(i)
        multd f4, f2, f0      ; multiplica a * X(i)
        ld f6, 0(r2)          ; carga Y(i)
        addd f6,f4,f6         ; suma a * X(i) + Y(i)
        sd 0(r2), f6          ; almacena Y(i)

iter2:
        ld f8, 8(r1)          ; carga X(i)
        multd f10, f8, f0     ; multiplica a * X(i)
        ld f12, 8(r2)         ; carga Y(i)
        addd f12,f10,f12      ; suma a * X(i) + Y(i)
        sd 8(r2), f12         ; almacena Y(i)

iter3:
        ld f14, 16(r1)        ; carga X(i)
        multd f16, f14, f0    ; multiplica a * X(i)
        ld f18, 16(r2)        ; carga Y(i)
        addd f18,f16,f18      ; suma a * X(i) + Y(i)
        sd 16(r2), f18        ; almacena Y(i)
        addi r1, r1, 24       ; incrementa índice X
        addi r2,r2, 24        ; incremente índice Y
        sgt r3, r1, r4        ; test por si finalizado
        beqz r3, iter1        ; bucle si no finalizado
        trap 0                ; Fin
```

Apéndice C

```
        .data
        .align 2

x:      .double 1.00, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09
        .double 1.10, 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19
        .double 1.20, 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29
        .double 1.30, 1.31, 1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39
        .double 1.40, 1.41, 1.42, 1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49
        .double 1.50, 1.51, 1.52, 1.53, 1.54, 1.55, 1.56, 1.57, 1.58, 1.59
        .double 1.60, 1.61, 1.62, 1.63, 1.64, 1.65, 1.66, 1.67, 1.68, 1.69
        .double 1.70, 1.71, 1.72, 1.73, 1.74, 1.75, 1.76, 1.77, 1.78, 1.79
        .double 1.80, 1.81, 1.82, 1.83, 1.84, 1.85, 1.86, 1.87, 1.88, 1.89
        .double 1.90, 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97, 1.98

y:      .double 2.00, 2.01, 2.02, 2.03, 2.04, 2.05, 2.06, 2.07, 2.08, 2.09
        .double 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19
        .double 2.20, 2.21, 2.22, 2.23, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29
        .double 2.30, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37, 2.38, 2.39
        .double 2.40, 2.41, 2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49
        .double 2.50, 2.51, 2.52, 2.53, 2.54, 2.55, 2.56, 2.57, 2.58, 2.59
        .double 2.60, 2.61, 2.62, 2.63, 2.64, 2.65, 2.66, 2.67, 2.68, 2.69
        .double 2.70, 2.71, 2.72, 2.73, 2.74, 2.75, 2.76, 2.77, 2.78, 2.79
        .double 2.80, 2.81, 2.82, 2.83, 2.84, 2.85, 2.86, 2.87, 2.88, 2.89
        .double 2.90, 2.91, 2.92, 2.93, 2.94, 2.95, 2.96, 2.97, 2.98

a:      .double 1.6180

        .text
        .global main

main:
        add r1,r0,x           ; R1 <= Dirección de comienzo de x
        add r2,r0,y           ; R2 <= Dirección de comienzo de y
        ld f0,a               ; F0 <= Valor de a
        addi r4,r1, 784       ; R4 <= Última posición del vector x

inicio:
        ld f2, 0(r1)          ; carga X(i)
        ld f8, 8(r1)          ; carga X(i)
        ld f14, 16(r1)        ; carga X(i)
        multd f4, f2, f0      ; multiplica a * X(i)
        multd f10, f8, f0     ; multiplica a * X(i)
        multd f16, f14, f0    ; multiplica a * X(i)
        ld f6, 0(r2)          ; carga Y(i)
        ld f12, 8(r2)         ; carga Y(i)
        ld f18, 16(r2)        ; carga Y(i)
        addd f6,f4,f6         ; suma a * X(i) + Y(i)
        addd f12,f10,f12      ; suma a * X(i) + Y(i)
        addd f18,f16,f18      ; suma a * X(i) + Y(i)
        sd 0(r2), f6          ; almacena Y(i)
        sd 8(r2), f12         ; almacena Y(i)
        sd 16(r2), f18        ; almacena Y(i)
        addi r1, r1, 24       ; incrementa índice X
        addi r2,r2, 24        ; incremente índice Y
        sgt r3, r1, r4        ; test por si finalizado
        beqz r3, inicio       ; bucle si no finalizado
        trap 0                ; Fin
```

Apéndice D

```
        .data
        .align 2

x:      .double 1.00, 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08, 1.09
        .double 1.10, 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19
        .double 1.20, 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29
        .double 1.30, 1.31, 1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39
        .double 1.40, 1.41, 1.42, 1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49
        .double 1.50, 1.51, 1.52, 1.53, 1.54, 1.55, 1.56, 1.57, 1.58, 1.59
        .double 1.60, 1.61, 1.62, 1.63, 1.64, 1.65, 1.66, 1.67, 1.68, 1.69
        .double 1.70, 1.71, 1.72, 1.73, 1.74, 1.75, 1.76, 1.77, 1.78, 1.79
        .double 1.80, 1.81, 1.82, 1.83, 1.84, 1.85, 1.86, 1.87, 1.88, 1.89
        .double 1.90, 1.91, 1.92, 1.93, 1.94, 1.95, 1.96, 1.97, 1.98

y:      .double 2.00, 2.01, 2.02, 2.03, 2.04, 2.05, 2.06, 2.07, 2.08, 2.09
        .double 2.10, 2.11, 2.12, 2.13, 2.14, 2.15, 2.16, 2.17, 2.18, 2.19
        .double 2.20, 2.21, 2.22, 2.23, 2.24, 2.25, 2.26, 2.27, 2.28, 2.29
        .double 2.30, 2.31, 2.32, 2.33, 2.34, 2.35, 2.36, 2.37, 2.38, 2.39
        .double 2.40, 2.41, 2.42, 2.43, 2.44, 2.45, 2.46, 2.47, 2.48, 2.49
        .double 2.50, 2.51, 2.52, 2.53, 2.54, 2.55, 2.56, 2.57, 2.58, 2.59
        .double 2.60, 2.61, 2.62, 2.63, 2.64, 2.65, 2.66, 2.67, 2.68, 2.69
        .double 2.70, 2.71, 2.72, 2.73, 2.74, 2.75, 2.76, 2.77, 2.78, 2.79
        .double 2.80, 2.81, 2.82, 2.83, 2.84, 2.85, 2.86, 2.87, 2.88, 2.89
        .double 2.90, 2.91, 2.92, 2.93, 2.94, 2.95, 2.96, 2.97, 2.98

a:      .double 1.6180

        .text
        .global main

main:
        add r1,r0,x           ; R1 <= Dirección de comienzo de x
        add r2,r0,y           ; R2 <= Dirección de comienzo de y
        ld f0,a               ; F0 <= Valor de a
        addi r4,r1, 784       ; R4 <= Última posición del vector x

inicio:
        ld f2, 0(r1)          ; carga X(i)
        ld f6, 0(r2)          ; carga Y(i)
        multd f4, f2, f0      ; multiplica a * X(i)
        ld f8, 8(r1)          ; carga X(i)
        ld f14, 16(r1)        ; carga X(i)
        ld f12, 8(r2)         ; carga Y(i)
        addi r1, r1, 24       ; incrementa índice X
        multd f10, f8, f0     ; multiplica a * X(i)
        addd f6,f4,f6         ; suma a * X(i) + Y(i)
        sgt r3, r1, r4        ; test por si finalizado
        ld f18, 16(r2)        ; carga Y(i)
        multd f16, f14, f0    ; multiplica a * X(i)
        addd f12,f10,f12      ; suma a * X(i) + Y(i)
        sd 0(r2), f6          ; almacena Y(i)
        sd 8(r2), f12         ; almacena Y(i)
        addd f18,f16,f18      ; suma a * X(i) + Y(i)
        sd 16(r2), f18        ; almacena Y(i)
        addi r2,r2, 24        ; incremente índice Y
        beqz r3, inicio       ; bucle si no finalizado
        trap 0                ; Fin
```