ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Universidad Rey Juan Carlos

PRÁCTICA 1
ARQUITECTURA DE COMPUTADORES

Integrantes del grupo: Alberto Martín Amengual, Álvaro Martínez Quiroga.

SECCIÓN 1:

Apartado 1, pseudocódigo: todos los nombres valores de tipo “int” no son registros si no nombres dados a las variables.

```
int main( ){
        vector a [10];
        vector b [10];
        vector c [10];
        int r1 = 0;
        int r4 = 0;
        int r20 = 72;
        int r19 = 8;
        int r18 = 10;
        int r17 = 1;
        int r16 = 2;
        int r5 = 40;
        while (r1 != r20){
                r1 = r1 + 8;
                r4 = r4 + 1;
                int r3;
                if (r1 < r5){
                        r3 = 1;
                }else{
                        r3 = 0;
                }
                if (r3 != r0){
                        r10 = a[0 + r1];
                        r11 = r10 + 10;
                        a[0 + r1] = r11;
                        r2 = r1 + 8;
                        r12 = b[0 + r2];
                        r12 = r12 + r16;
                        r12 = r12 + r11;
                        b[0 + r1] = r12;
                        r13 = r12 + r4;
                        c[0 + r1] = r13;
                }else{
                        r6 = r1 – r19;
                        r10 = a[0 + r6];
                        r11 = r10 – r18;
                        a[0 + r1] = r11;
                }
        }
        r10 = a[r0];
        r11 = r10 + 5;
        a[r0] = r11;
        b[r0] = r17;
        c[r0] = r17;
        return 0;
}
```

Línea en **rojo**: carga en variable el valor de la posición a la que se accede en ese momento del vector correspondiente.
Línea en **verde**: guarda un valor de una variable en la posición a la que se accede al vector en ese momento.

Apartado 2, paradas:

CICLO	TIPO EXCEPCIÓN	INSTRUCCIÓN CAUSA	DATO CAUSANTE
12	RAW	slt r3, r1, r5	r1
14	RAW	beq r3, r0, else	r3
18	RAW	daddi r11, r10 10	r10
21	RAW	sd r11, a(r1)	r11
25	RAW	ld r12, b(r2)	r2
28	RAW	dadd r12, r12, r16	r12
31	RAW	dadd r12, r11, r12	r12
34	RAW	sd r12, b(r1)	r12
38	RAW	sd r13, c(r1)	r13
45	RAW	slt r3, r1, r5	r1
47	RAW	beq r3, r0, else	r3
51	RAW	daddi r11, r10 10	r10
54	RAW	sd r11, a(r1)	r11
58	RAW	ld r12, b(r2)	r2
61	RAW	dadd r12, r12, r16	r12
64	RAW	dadd r12, r11, r12	r12
67	RAW	sd r12, b(r1)	r12
71	RAW	sd r13, c(r1)	r13
78	RAW	slt r3, r1, r5	r1
80	RAW	beq r3, r0, else	r3
84	RAW	daddi r11, r10 10	r10
87	RAW	sd r11, a(r1)	r11
91	RAW	ld r12, b(r2)	r2
94	RAW	dadd r12, r12, r16	r12
97	RAW	dadd r12, r11, r12	r12
100	RAW	sd r12, b(r1)	r12
104	RAW	sd r13, c(r1)	r13
111	RAW	slt r3, r1, r5	r1
113	RAW	beq r3, r0, else	r3
117	RAW	daddi r11, r10 10	r10
120	RAW	sd r11, a(r1)	r11
124	RAW	ld r12, b(r2)	r2
127	RAW	dadd r12, r12, r16	r12
130	RAW	dadd r12, r11, r12	r12
133	RAW	sd r12, b(r1)	r12
137	RAW	sd r13, c(r1)	r13
144	RAW	slt r3, r1, r5	r1
146	RAW	beq r3, r0, else	r3
151	RAW	ld r10, a(r6)	r6
154	RAW	dsub r11, r10, r18	r10
157	RAW	sd r11, a(r1)	r11
164	RAW	slt r3, r1, r5	r1
166	RAW	beq r3, r0, else	r3
171	RAW	ld r10, a(r6)	r6
174	RAW	dsub r11, r10, r18	r10
177	RAW	sd r11, a(r1)	r11
184	RAW	slt r3, r1, r5	r1
186	RAW	beq r3, r0, else	r3
191	RAW	ld r10, a(r6)	r6

194	RAW	dsub r11, r10, r18	r10
197	RAW	sd r11, a(r1)	r11
204	RAW	slt r3, r1, r5	r1
206	RAW	beq r3, r0, else	r3
211	RAW	ld r10, a(r6)	r6
214	RAW	dsub r11, r10, r18	r10
217	RAW	sd r11, a(r1)	r11
224	RAW	slt r3, r1, r5	r1
226	RAW	beq r3, r0, else	r3
231	RAW	ld r10, a(r6)	r6
234	RAW	dsub r11, r10, r18	r10
237	RAW	sd r11, r10, r18	r10
242	RAW	daddi r11, r10, 5	r10
245	RAW	sd r11, a(r0)	r11

Apartado 3, ejecución con habilitación de "forwarding":

- Ciclos: 159.
- CPI: 1,336 ciclos por instrucción.
- Speedup: CiclosAntiguo / CiclosNuevo = 253 / 159 = 1,59.
- Se obtienen menos ciclos que en el caso anterior gracias a la activación de la posibilidad de adelantamiento de datos y el rendimiento mejora ya que todos los riesgos que ocurren son de tipo RAW, lo que implica que en muchos casos se podrán ahorrar esos ciclos de parada que se producían.

Apartado 4, mejora del código:

```
.data
A:      .word 10,11,12,13,14,15,16,17,18,19
B:      .word 20,21,22,23,24,25,26,27,28,29
C:      .word 30,31,32,33,34,35,36,37,38,39

        .text

        daddi R1,R0,0
        daddi R4,R0,0
        daddi R20,R0,72
        daddi R19,R0,8
        daddi R18,R0,10
        daddi R17,R0,1
        daddi R16,R0,2
        daddi R5,R0,40

bucle:  daddi R1,R1,8
        daddi R4,R4,1
        slt R3,R1,R5
        beq R3,R0,else
        ld R10,A(R1)
        daddi R2,R1,8
        daddi R11,R10,10
        ld R12,B(R2)
        sd R11,A(R1)
        dadd R12,R12,R16
        dadd R12,R11,R12
```

```
        dadd R13,R12,R4
        sd R12,B(R1)
        sd R13,C(R1)
        bne R1,R20,bucle
        j fin

else:   dsub R6,R1,R19
        ld R10,A(R6)
        dsub R11,R10,R18
        sd R11,A(R1)
        bne R1,R20,bucle
fin:    ld R10,A(R0)
        sd R17,B(R0)
        daddi R11,R10,5
        sd R17,C(R0)
        sd R11,A(R0)
        halt
```

En este caso los únicos cambios que se pueden realizar es cuando realiza el bucle, las cuatro primeras instrucciones no se pueden modificar, ya que son dependientes entre las cuatro, y lo mismo con la parte del "else".

Hemos marcado las instrucciones que suben en **amarillo** y las que bajan en **rojo**.

- El primer error que solucionamos es el RAW de ld R10, A(R1) y daddi R11, R10, 10 por R10, en este caso pasaríamos de tener 2 paradas a 1.

- El segundo sería el RAW que se produce entre la instrucción daddi R11, R10, 10 y sd R11, A(R1), que como en el caso anterior pasamos de tener 2 paradas a 1.

- El tercero sería el RAW que se produce entre las instrucciones dadd R13, R12, R4 y sd R13, C(R1) por R13, por R13 y en este caso pasaríamos de tener 2 paradas a 1.

- Otro error que evitamos seria daddi R2, R1, 8 y ld R12, B(R2) que al realizar una parada en la instrucción daddi R11, R10, 10, evitamos las dos paradas que se realiza sin reordenación de código.

- El último error que evitamos es el RAW que se produce entre la instrucción daddi R11, R10, 10 y sd R11, A(R1), como en el caso anterior y al haber una parada en la instrucción ld R12, B(R2), evitaríamos 2 paradas.

- En el último cambio reordenamos la parte de guardado de variables, consiguiendo así evitar paradas por las instrucciones daddi R11,R10,5 y sd R11,A(R0), por R11.

Por lo que, en total, pasaríamos de tener 17 paradas de tipo RAW en el primer bucle a 10. Finalmente obtendríamos:

- Ciclos: 150.
- CPI: 1,261 ciclos por instrucción.
- 14 paradas producidas por RAWs.

Apartado 5, ejecución activando "Delay Slot", máximo número de mejoras:

- Ciclos: 132.
- CPI: 1,039 ciclos por instrucción.
- 1 parada producidas por RAWs.

SECCIÓN 2:

Apartado 6,

a) Las paradas producidas son por la primera iteración son:

CICLO	TIPO EXCEPCIÓN	INSTRUCCIÓN CAUSA	DATO CAUSANTE
10	RAW	andi r9, r3,1	r3
12	RAW	dsub r9,r0,r9	r9
16	RAW	and r6,r4,r9	r9
18	RAW	daddu r2,r2,r6	r6
21	RAW	sltu t7,r2,r6	r2
24	RAW	dsllv r7,r7,r5	r7
28	RAW	dsllv r10,r10,r5	r10
32	RAW	or r2,r2,r7	r2
36	RAW	dsub r9,r0,r9	r9
40	RAW	or r3,r3,r10	r3
44	RAW	bnez r1,again	r1

En total, 20 paradas por RAW.

Tras realizar la primera iteración las paradas realizadas son:

CICLO	TIPO EXCEPCIÓN	INSTRUCCIÓN CAUSA	DATO CAUSANTE
49	RAW	daddu r2,r2,r6	r6
52	RAW	sltu t7,r2,r6	r2
55	RAW	dsllv r7,r7,r5	r7
59	RAW	dsllv r10,r10,r5	r10
63	RAW	or r2,r2,r7	r2
67	RAW	dsub r9,r0,r9	r9
71	RAW	or r3,r3,r10	r3
75	RAW	bnez r1,again	r1

En este caso esas instrucciones se repiten durante 63 iteraciones. En total habría 64 iteraciones en el bucle.

En total obtendríamos:

- Ciclos: 2008.
- CPI: 2,204 ciclos por instrucción.
- 1028 paradas producidas por RAWs.

b) ¿Por qué crees que la segunda instrucción del código es una instrucción de NOP que no realiza ningún trabajo?

La instrucción nop sirve para rellenar el cauce de ejecución como una normal, en este caso se utiliza, ya que la instrucción jal no sabe si salta hasta la que no llega a la fase de ejecución (X o Ex), por lo que se utiliza la función nop, para rellenar ese cauce y así no comenzar a ejecutar directamente a la instrucción halt.

Apartado 7,

a) Las paradas producidas activando adelantamiento "forwarding":

CICLO	TIPO EXCEPCIÓN	INSTRUCCIÓN CAUSA	DATO CAUSANTE
26	RAW	bnez r1,again	R1

Activando adelantamiento solo existe una parada por RAW por lo que al terminar de ejecutar obtendremos 64 paradas por RAW frente a los 1028 que obteníamos sin adelantamiento.

En total obtendríamos:

- Ciclos: 1044
- CPI: 1,146 ciclos por instrucción.
- 64 paradas producidas por RAWs.

En caso de que reordenáramos código, solo tendríamos que solucionar una instrucción, por lo tanto, el código quedaría de la siguiente manera:

```
        .data
x:      .word 0xFFFFFFFFFFFFFFFF
y:      .word 0xFFFFFFFFFFFFFFFF
w:      .word 0,0

        .text

start:  jal mul
        nop
        halt
mul:    daddi r1,r0,64
        daddi r5,r0,63
        daddu r2,r0,r0
        daddu r10,r0,r0
        ld r3,x(r0)
        ld r4,y(r0)
        andi r9,r3,1
        dsub r9,r0,r9
        dsrl r3,r3,1
again:  and r6,r4,r9
        daddu r2,r2,r6
        sltu r7,r2,r6
        dsllv r7,r7,r5
        andi r10,r2,1
        dsllv r10,r10,r5
        dsrl r2,r2,1
        or r2,r2,r7
        andi r9,r3,1
        dsub r9,r0,r9
        dsrl r3,r3,1
        daddi r1,r1,-1
```

```
or r3,r3,r10
bnez r1,again

sd r2,w(r0)
sd r3,w+8(r0)
jr r31
```

Hemos marcado las instrucciones que suben en **amarillo** y las que bajan en **rojo**.

En este caso al cambiar la instrucción daddi r1, r1, -1 por or r3, r3, r10 evitaríamos la parada que se produce en bnez r1, again por r1, por lo que no habría ninguna parada en todo el código.

Con esta reordenación de código y con adelantamiento este programa:

- Ciclos: 980.
- CPI: 1,076 ciclos por instrucción.
- 0 paradas producidas por RAWs.

Apartado 8, ejecución activando "Delay Slot", máximo número de mejoras:

- Ciclos: 980.
- CPI: 1,004 ciclos por instrucción.
- 0 paradas producidas por RAWs.

SECCIÓN 3:

Apartado 9, ejecutando hail.s obtenemos los siguientes resultados (la entrada es = 100):

- Ciclos: 644.
- CPI: 2,205 ciclos por instrucción.
- 274 paradas por RAWs.

Posibles soluciones para evitar paradas:

1. Respecto a las paradas por RAW: reordenación de código y uso de adelantamientos.
2. Respecto a los saltos: habilitar la condición de salto sí tomado.

Apartado 10, mejora de código:

```
.text
    lwu r8,DATA(r0)
    lwu r9,CONTROL(r0)

    daddi r11,r0,4

    daddi r1,r0,titulo
    sd r1,(r8)
    sd r11,(r9)

    daddi r1,r0,pregunta
    sd r1,0(r8)
    sd r11,0(r9)

    daddi r1,$zero,8
    sd r1,0(r9)
    ld r1,0(r8)
    sd r1,max(r0)
```

```
bucle:   andi r3,r1,1
         daddi r12,r0,1
         beqz r3,par

impar:   daddu r2,r1,r1
         dadd r1,r2,r1
         daddi r1,r1,1
         j maximo

par:     dsrl r1,r1,1

maximo: sd r1,(r8)
         ld r4,max(r0)
         sd r12,(r9)
         slt r3,r4,r1
         beqz r3,fin
         sd r1,max(r0)

fin:     slti r3,r1,2
         beqz r3,bucle

         ld r2,max(r0)
        daddi r1,r0,str
        sd r1,(r8)
        sd r11,(r9)
        sd r2,(r8)
        sd r12,(r9)
        halt
```

Hemos marcado las instrucciones que suben en **amarillo** y las que bajan en **rojo**.

- En este caso para el RAW que se produce al inicio del bucle entre andi r3, r1, 1 y beqz r3, par por r3, lo evitamos añadiendo daddi r12, r0,1 entre medias, en este caso no evitaríamos tener menos ciclos a la hora de hacer el bucle, pero evitaríamos 1 parada por RAW cada vez que realizamos el bucle.

- La segunda parada que evitamos es en este caso un RAW entre las instrucciones ld r4, max(r0) y slt r3, r4, r1 por r4, bajando la instrucción sd r12, (r9).

Con esta reordenación de código y con adelantamiento este programa:

- Ciclos: 444.
- CPI: 1,405 ciclos por instrucción.
- 50 paradas por RAWs.

Apartado 11, activando “Delay Slot”, máximo número de mejoras:

- Ciclos: 332.
- CPI: 1,137 ciclos por instrucción.
- 36 paradas por RAWs.