Computer Architecture and Engineering - 3rd Course. **Sol. of Examen 1st partial** **24/1/2017**

Answer each question in a different answer sheet. Available time: 2h 30'

1. **(3 points)**

   A program is sequentially executed in 100 seconds on a computer. It devotes 20 % of this time to carry out arithmetic operations and 50 % loading data from, and storing results to, memory. During the remaining 30 % of the time, the programa carries out other tasks. It is worth mentioning that half of the time devoted to these other tasks concentrates in tasks whose execution cannot be parallelized. Answer the following questions:

   a) How long would the execution of the program take if the program runs on a computer providing SIMD (*Single Instruction, Multiple-Data*) instructions suitable for processing 4 elements in each ALU, load and store operation? Asume all concerned operations can be replaced by SIMD instructions.

   b) How long would the execution of the program take if using a 10-core, without SIMD instructions support, computer?

   c) If each of the 10 cores available provides support to the considered SIMD instructions, which will be the resulting speed-up with respect to the original computer? and, which will be the new execution time of the considered program?

   d) Which will be the best possible performance (speed-up and execution time for the considered program) if one could integrate in the computer as many cores as desired?

   Solución:

   a) El 70 % del tiempo, el computador ejecuta operaciones aritméticas y de almacenamiento/lectura de datos en memoria, aplicando Amdahl: $\frac{100}{t_{SIMD}} = \frac{1}{0{,}3+\frac{0{,}7}{4}}$
   Con lo que se deduce que el tiempo de ejecución (t_{SIMD}) se reduce a 47,5s.

   b) El 85 % del tiempo, el computador ejecuta operaciones que pueden paralelizarse. Por tanto $\frac{100}{t_{MIMD}} = \frac{1}{0{,}15+\frac{0{,}85}{10}}$
   Con lo que se deduce que el tiempo de ejecución (t_{MIMD}) se reduce a 23,5s.

   c) Las dos mejoras pueden aplicarse durante el 70 % del tiempo, $S_{local_1} = 10 \times 4 = 40$, mientras que durante el 15 % del tiempo sólo es posible obtener beneficio de los 10 cores disponibles, pero no del soporte a la ejecución de instrucciones SIMD, con lo que $S_{local_2} = 10$.
   Planteando que $S = \frac{1}{0{,}15+\frac{0{,}7}{40}+\frac{0{,}15}{10}} = 5,48$, con lo que el nuevo tiempo de ejecución será de 18,25s.

   d) Si dispusiéramos de un número infinito de núcleos, calcularíamos la aceleración máxima de la siguiente forma: $S = \frac{1}{0{,}15} = 6,67$, con lo que el mejor tiempo de ejecución posible en ese caso sería de 15s.

   □

2. **(2.5 points)**

   The execution of a program shows that 10 % ($CPI = 1{,}5$) of instructions are branches, 30 % ($CPI = 2$) are loads and stores, 10 % ($CPI = 1$) are ALU, 20 % ($CPI = 4$) are single precision floating point instructions and finally, the remaining 30 % ($CPI = 6$) are double precision floating point instructions.

   Answer the following questions:

   a) Average execution CPI.

   b) The use of vector instructions (SIMD) is introduced in order to improve the execution of floating point instructions. These instructions enable:

   - The combination of 4 single-precision floating point instructions into a single SIMD instruction.
   - The combitnation of 2 double-precision floating point instructions into a single SIMD instruction.

   On the other hand, it is known that SIMD instructions have the same CPI that the corresponding floating point ones.

   Assuming that all floating point instructions in the considered application can be replaced by SIMD instructions, which will be the new distribution of instructions?

c) Which will be the execution speed up resulting from the use of SIMD instructions?

Solución:

a) CPI medio de la ejecución

$$CPI = 0{,}1 \times 1{,}5 + 0{,}3 \times 2 + 0{,}1 \times 1 + 0{,}2 \times 4 + 0{,}3 \times 6 = 3{,}45.$$

b) Nueva distribución de instrucciones.

Tipo	Ratio
saltos	10/70
carga/alm.	30/70
arit. enteras	10/70
SIMD simple prec.	(20/4)/70 = 5/70
SIMD doble prec.	(30/2)/70 = 15/70
Total	70/70

c) Aceleración de la mejora.
El tiempo de ejecución de la aplicación original es $T = I \times CPI \times t = I \times 3{,}45 \times t$, mientras que el de la aplicación con las instrucciones SIMD es $T_{SIMD} = I_{SIMD} \times CPI_{SIMD} \times t$.
El número de instrucciones con SIMD es

$$I_{SIMD} = \frac{70}{100} \times I = 0{,}7 \times I.$$

El CPI con SIMD es

$$CPI_{SIMD} = \frac{10}{70} \times 1{,}5 + \frac{30}{70} \times 2 + \frac{10}{70} \times 1 + \frac{5}{70} \times 4 + \frac{15}{70} \times 6 \approx 2{,}79.$$

Luego,

$$T_{SIMD} = 0{,}7 \times I \times 2{,}79 \times t \approx 1{,}95 \times I \times t.$$

Por tanto la aceleración es $3{,}45/1{,}95 \approx 1{,}77$.

□

3. **(2.5 points)** A MIPS processor is available. It integrates the following operators:

   - Pipelined Adder/Substracter. Lat= 2, IR= 1.
   - Pipelined Multiplier. Lat= 4, IR= 1.
   - Conventional Divider. Lat= 5, IR= $\frac{1}{5}$.

   Structural and data hazards are detected in stage ID, thus inserting as many stalls are necessary and using shortcircuits whenever possible. Control hazards are solved through delayed branching, taking into account that conditions and branch destination addresses are computed during ID and the PC is updated with the destination address at the end of EX.

   The code executed in the considered processor is:

```
        dadd r1, r0, x
        dadd r2, r0, z
        dadd r4, r1, 32
        l.d f0, a(r0)           ; Load a
loop:
        l.d f1, 0(r1)
        div.d f2, f1, f0
        mult.d f3, f1, f0
        add.d f4, f2, f3
```

```
        s.d f4, 0(r1)
        dadd r1, r1, 8
        dsub r5, r4, r1
        bnez r5, loop
        nop
        nop
        nop
        trap 0.
```

Answer the following questions using the notation: `IF` instruction fetch, `ID` instruction decoding, `EX` monocycle execution, `A1,A2` adder/substracter execution stages, `M1,M2,M3,M4` multiplier execution stages, `D1,D2,D3,D4,D5` divider execution stages, `ME` memory access stage and `WB` write back to registers. Multicycle instructions do not carry out the `ME` stage.

a) Complete the instructions-time diagram for the first loop iteration including until the first instruction of the second loop iteration.

b) Complete the instructions-time diagram of the last loop iteration, from the *dsub* instruction to the *trap* instruction (both included).

c) Compute the average CPI of a loop iteration.

d) Compute the TOTAL execution time of the provided code in cycles for n iterations (from the fetch of the first instruction dadd r1,r0,x until the WB stage of instruction trap 0).

Solución:

a) Dibuje el diagrama instrucciones-tiempo perteneciente a la primera iteración del bucle incluyendo hasta la primera instrucción de la segunda iteración.

```
                        1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
        dadd r1, r0, x  IF ID EX ME WB
        dadd r2, r0, z     IF ID EX ME WB
        dadd r4, r1, 32       IF ID EX ME WB
        l.d f0, a(r0)            IF ID EX ME WB
loop:   l.d f1, 0(r1)               IF ID EX ME WB
        div.d f2, f1, f0            |  IF id ID D1 D2 D3 D4 D5 WB
        mult.d f3, f1, f0           |     if IF id ID M1 M2 M3 M4 WB
        add.d f4, f2, f3            |           if IF id id id ID A1 A2 WB
        s.d f4, 0(r2)               |                 if if if IF ID EX ME WB
        dadd r1, r1, 8              |                             IF ID EX ME WB
        dsub r5, r4, r1             |                                IF ID EX ME WB
        bnez r5, loop               |                                   IF id ID EX ME WB
        nop                         |                                      if IF ID EX ME WB
        nop                         |                                            IF ID EX ME WB
loop:   l.d f1, 0(r1)               |                                               IF ID EX ME WB
                                    |---------------- 16 CICLOS -------------------|
        trap 0
```

b) Si cambiamos el salto retardado por un predictor BTB *Branch Target Buffer* de un bit, ¿cuál será el CPI de la segunda iteración del bucle? Justifique la respuesta.
$CPI = \frac{16\ ciclos}{10\ instrucciones} = 1{,}6$

c) Calcule el tiempo de ejecución TOTAL del código proporcionado (en ciclos) para n iteraciones.
$t_{ejec} = ciclos\ iniciales + (n\ iteraciones \times ciclos\ por\ iteracion) + ciclos\ finales$
$= 4 + (n \times 16) + 5 = 9 + 16 \times n$ ciclos

□

4. **(2 points)** A processor with an MIPS-like instruction set is available. It integrates an execution unit pipelined in the following stages:

   - **IF** Instruction Fetch.
   - **ID** Instruction decoding and register reading.
   - **EX** ALU execution. Computation of conditions and destination addresses for branches. Computation of the address for a memory (load/store) access.

- **ME** Memory access.
- **WB** Writing back of results to the register file.

The following table shows the behavior of a *Branch Target Buffer* predictor when it predicts that a conditional branch instruction is **taken** and it is finally **not taken**.

I. Salto	IF	ID	EX	ME	WB			
DEST		IF	ID	X				
DEST+1			IF	X				
PC+1				IF	ID	EX	ME	WB

Answer the following questions:

a) In which stage is updated the PC?

b) In which stage is the prediction carried out?

c) Complete the same tables considering the following cases: i) Prediction is **not taken** and it is **not taken** , ii) Prediction is **not taken** and it is **taken** and iii) Prediction is **taken** and it is **taken**

Solución:

a) ¿En que fase se realiza la escritura del PC?
En la fase EX

b) ¿En que fase se realiza la predicción?
En la fase ID

c) Suponiendo que la predicción y la escritura del PC se realizan en las mismas fases que en el caso del BTB, rellene las mismas tablas para el predictor *Branch Prediction Buffer*

Predice que **no** salta y **no** salta.

I. Salto	IF	ID	EX	ME	WB			
PC+1		IF	ID	EX	ME	WB		
PC+2			IF	ID	EX	ME	WB	
PC+3				IF	ID	EX	ME	WB

Predice que **no** salta y **sí** salta.

I. Salto	IF	ID	EX	ME	WB			
PC+1		IF	ID	X				
PC+2			IF	X				
DEST				IF	ID	EX	ME	WB

Predice que **si** salta y **no** salta.

I. Salto	IF	ID	EX	ME	WB				
PC+1		IF	X						
DEST			IF	X					
PC+1				IF	ID	EX	ME	WB	
PC+2					IF	ID	EX	ME	WB

Predice que **si** salta y **sí** salta.

I. Salto	IF	ID	EX	ME	WB				
PC+1		IF	X						
DEST			IF	ID	EX	ME	WB		
DEST+1				IF	ID	EX	ME	WB	
DEST+2					IF	ID	EX	ME	WB

□