Universidad Carlos III de Madrid

Computer Architecture

Group 89

Final exam 2012-2013

Name:_____

NIA: _____

I. (10 points) Answer the following questions. ***A correct answer scores 0.9 points, while an incorrect answer subtracts 0.3 points.***

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Answer	E	C	C	B	A	C	BG	B	A	D	B

1. What is not an advantage of Open Flow compared with a „normal network“?
a) easy monitoring of the network b) controlling the network over one central instance b) special routes for special types of packets d) uniform network utilization e) all of them
2. Which is NOT the feature of Storage Class Memory? a) High DRAM-like reading performance b) Large capacity c) Volatile d) Relatively low cost e) None of the above
3. The main difference between DDR modules and HMC is: a) HMC is for permanent storage meanwhile DDR is volatile. b) They are both more or less the same but HMC has more latency. c) DDR is based in a two dimensional fashion-way and HMC uses the three dimensions to pile one chip on top of the other.
4. What are the advantages of ARM architecture? a) Performance, available software, power cost b) Power cost, size, innovation c) Size, price-per-unit, configurability
5. Inexact processors have an advantage over regular ones in terms of: a) efficiency and size b) accuracy and efficiency c) cost, accuracy and efficiency d) number of cores, cost and efficiency
6. Memristors can be used to improve the performance of: a) Non-volatile memory. b) Volatile memory. c) Both d) None of them.
7. The two most important criteria that Green500 takes into account to create its list are (***choose two***): a) scalability b) energy c) responsiveness d) availability e) graphics performance f) price g) performance.
8. Which of the following is NOT true about Solid State Machine: a) Use flash memory b) Store data on spinning platter c) Limited storage capacity d) Less noise and heat

9. Which is the brand new feature introduced in Blue Gene Q? a) Transactional Memory b) Reduction in the number of cores c) IBM's 45 nm SOI technology
10. What is the key objective of the Mont-Blanc project? a) To create the most powerful supercomputer in the world. b) To build a complete software stack for the supercomputer. c) To test and introduce Nvidia ARM-CUDA processors in the HPC world. d) To create an energy-efficient exascale supercomputer.
11. Intel’s Many-Core coprocessors do not take advantage of SIMD (Single Instruction Multiple Data) units, while Nvidia’s GPU coprocessors do. a) True b) False

II. (20 points) Shortly answer the following questions:

a) Explain what is fairness in the implementation of a lock. Give one example of unfair implementation and one example of fair implementation.

b) Explain Early Restart and Critical Word First cache optimization.

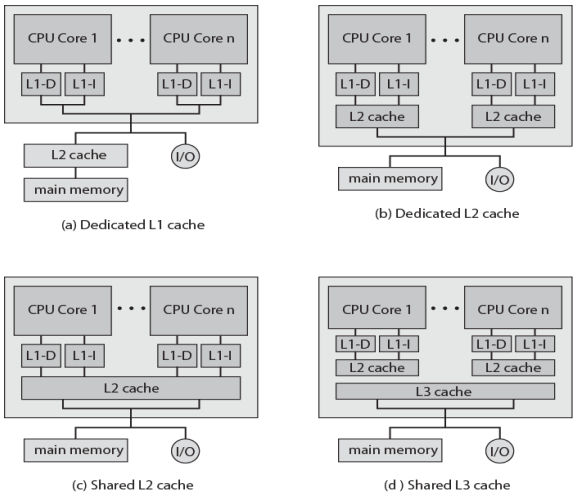
c) What is the difference between snoopy-based and directory-based cache coherence?

d) Explain shortly what is release consistency by answering the following questions: Which program orders are relaxed? What are the constraints? What is the motivation of taking this approach?

Solutions:

Slides

III. (20 points) One computer has a Intel Dual Core CPU as shown in the figure from below. Each core has a local data cache (L1-D) and a local instruction cache (L1-I). Both cores share a L2 cache. The coherence protocol is MSI. For all caches, a cache line has 16 bytes and one integer occupies 4 bytes.



The caches have the following characteristics

Type	Size	Access time	Write policy
L1I	16KB	1 ns	*write-through*
L1D	32KB	1 ns	*write-through*
L2	2 MB	5 ns	*write-back*
Memory	4GB	100 ns	--

Both cores start to execyte the code from below at the same time. Note that, because of sleeps operations, the for loops are executed at different times. T0, T1, T2, T3, T4 y T5 label diferent execution times of the code. Note that T0<T1<T2<T3<T4<T5.
Assume that:

- The variables **i, j, tmp1** and **tmp2** are stored in registers.
- The vector **a** is in the main memory but not in cache.
- Each element of a is an integer and ocupies 4 bytes.
- Execution time is so small that is can be neglected.

```
//Code executed on core 1
sleep (2 seconds)
for (i=0;i<40;i++){
     tmp1=tmp1+a[i];}
sleep (3 seconds)
for (j=0;j<40;j++){
```

T0

el

```
//Code executed on core 2
for (i=0;i<40;i++){
     tmp2=tmp2+a[i];}
sleep (10 seconds)
for (j=0;j<40;j++){
     tmp2=tmp2+a[j];}
```

El núcleo 1 ejecuta siguiente programa:

T1

```
     for (i=0;i<100;i++){
          tmp=tmp+a[i];
}
```

T2

```
          sleep (3 segundos)
     for (i=0;i<100;i++)
               a[i]=i;
          }
```

T3

T4

```
     {
```

T5

You are required to:

1.
Indicate the MSI states of the cache lines storing a[0] for both cores at times T0, T1, T2, T3, T4 y T5. Justify your answer.

	T0	T1	T2	T3	T4	T5
Core 1						
Core 2						

2.
Calculate the time to execute the code of the core 2. Consider only the data accesses, in other words consider that the times for instruction access and executing the loop are 0.

SOLUCIÓN:

1.- El bloque a[0] está inicialmente en memoria. Durante la primera iteración del primer bucle, se accede a a[0] y se produce un fallo caché en L1D del núcleo 1 que se propaga a L2, produciendo un acierto. El bloque es transferido de la caché L2 a la L1D del núcleo 2. El estado del bloque pasa

de Inválido a Compartido (I -> S). Posteriormente, tras 3 segundos, el núcleo 1 escribe en a[0] pasando de estado compartido a modificado (S->M) e invalidando la copia del núcleo 2. En t=5 segundos, el núcleo 2 lee a[0] pasando de estado modificado a compartido (M->S).

	T0	T1	T2	T3	T4	T5
Estado de a[0] en L1D del núcleo 1	I	I	S	M	M	S

2.- El bloque a[0] está inicialmente en memoria. Durante la primera iteración del primer bucle, se accede a a[0] y se produce un fallo caché en L1D del núcleo 2 que se propaga a L2 produciendo otro fallo. El bloque es transferido de memoria a la caché L2 y de allí a la L1D del núcleo 1. El estado del bloque pasa de Inválido a Compartido (I -> S). Posteriormente, en t=5 segundos, el núcleo 1 escribe en a[0] invalidando la copia del núcleo 2 (pasando de compartido a inválido S->I). En t=10 segundos, el núcleo 2 lee a[0] pasando de estado modificado a compartido (M->S).

	T0	T1	T2	T3	T4	T5
Estado de a[0] en L1D del núcleo 1	I	S	S	I	I	S

2.-
Primer bucle (índice i):
Cuando se ejecuta el primer bucle del núcleo 2 las entradas de a están en la memoria principal. El número de entradas por bloque es de 16/4=4 y el número de bloques accedidos es 40/4=10.

IV. (20 points) The following MIPS code reads and writes the same memory address several times. Assume that initially R1=1 and R2=1000.

```
            ADDI  R3,R3,1
Loop:       LD    R4,0(16)
            MULTD R5,R4,R4
            ADDD  R5,R3,R5
            SD    R5,0(16)
            DADDI
      R3,R4,1
```

Assume that the code executes on a MIPS processor having the following phases: *fetch*, decode, execution, memory and write-back. The processor issues one instruction per cycle and has *forwarding.* All instructions require one cycle to execute except SD, LD, ADDD and MULTD, which require two cycles. Assume that the branch prediction strategy is ***not-taken***.

You are asked to:

a) Indicate all data dependencies.
b) Indicate the instructions that cannot be reordered.
c) Show in the table from below how the first program iteration proceeds through the pipeline assuming *forwarding.*

d) How many cycles requires the ***whole*** program to execute?
e) How many cycles requires the ***whole*** program to execute if the branch prediction strategy is changed to **taken**?

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ADDI	F	D	E	M	W														
LD																			
MULTD																			
ADDD																			
SD																			
DADDI																			
DADDI																			
BNE																			

SOLUCION

- WAW: I4 con I3 en R5
- W*A*W: I1 con I6 en R3

Se obtendría un resultado incorrecto si I4 se ejecuta antes de I3 En aquellos procesadores segmentados donde las instrucciones pueden ser reordenadas.

a) Diagrama de temporización asumiendo *forwarding.* Describa las diferencias existentes entre esta técnica y cuando no se usa *forwarding*.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ADDI	F	D	E	M	W												
LD		F	D	E	M	W	W										
MULTD			F	D	-	E	E	M	W								
ADDD				F	-	D	-	E	E	M	W						
SD						F	-	D	D	E	M	W					
DADDI								F		D	E	M	W				
DADDI										F	D	E	M	W			
BNE											F	-	D	E	M	W	
Inext													F(PC+4)	FI1			

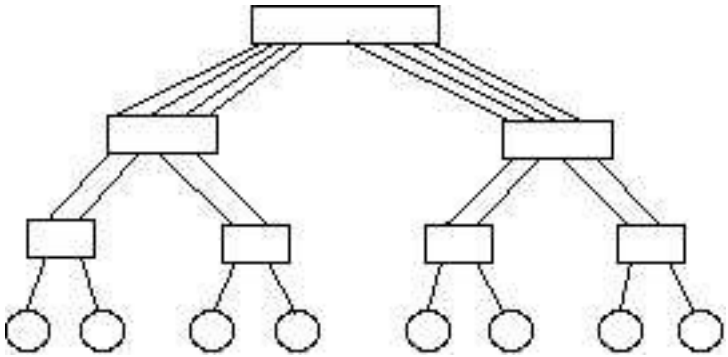
Vemos en sombreado los ciclos adicionales de las operaciones LD, ADDD, MULTD y SD.

b) Número de ciclos que tardaría el programa en ejecutarse.
Número de ciclos antes del bucle=1
Número de ciclos por iteración del bucle=12
Número de ciclos extra de la última instrucción del bucle=3

Iteraciones del bucle=1000
Número de ciclos=Número de ciclos antes del bucle+(Número de ciclos por iteración del bucle* Iteraciones del bucle)+Número de ciclos extra
1+(12*1000)+3=12004 ciclos.

V. (15 points) For the fat tree topology from below assume that the overhead of sending/receiving a message is 1 µs, the bandwidth of all links is 16 MB/s, the routing overhead is 100 ns per hop.

a) Compute the diameter of the network.
b) Compute the **maximal** time it takes a message of 64 bytes to travel between two nodes of a fat tree with 8 nodes for the following routing strategies: i) Store and forward ii) Cut-through
c) Give **two** reasons for having more links in the upper part of the hierarchy.



Solution:
a) 6
b) Maximal distance h=6
i) o+h(n/b + r)
ii) o+n/b + h r
c) Congestion and fault tolerance

VI. (15 points) Assume that a lock is implemented in the following way:

```
lock(location) {
      acquired = 0;
      while (! acquired) { /* lock is alredy acquired */
            acquired = test&set(location);
            if (!acquired) /* test&set didn't succeed*/
                  sleep (2ms); /* sleep some time
      }
}

unlock(location) { location = 0; }
```

Four processes try to acquire the lock at the same time by executing the following code:

```
lock(location);
do some computation for a time of 3ms
unlock(location).
```

Answer the following questions:

a) After how much time in ms will acquire the lock each of the four processes? Assume that the overhead of the coherency protocol is zero and executing only one instruction takes a negligible time.

b) Is the lock implementation efficient? Justify your answer.

c) Can the lock implementation be improved?

d) Is the order in which processes call the lock the same as the order in which they acquire the lock?

Solution:

a) P1: 0 P2: 4ms P3: 8ms P4: 12 ms

b) It is not efficient. The waiting processes sleep more than nessary. Even though the lock becomes available after 3ms, the waiting process sleep one more ms which remains unused.

c) Use exponential backoff for sleeping

d) No, the lock implementation is unfair.