Computer Architecture Master Sample Paper

Q1

RISC-1

Explain the operation of the RISC-1 register windows & explains how its design strives to achieve a single cycle procedure call & return, and delayed jumps. Drawing a diagram of a typical procedure stack frame, explains how global variables, local variables & procedure parameters are accessed and describe how registers are saved and restored across procedure calls.

Why is a register window overflow and underflow mechanisms needed?
Explain in detail the conditions under which register window underflow and overflow occur and the steps taken to resolve them.

Compare the RISC-1 approach with that of the IA32 CPU. Use diagrams to illustrate your answer

What is tail recursion optimisation and what are its advantages? Show how tail recursion optimisation could be applied to your code.

IA32

Briefly describe the IA32 procedure calling conventions. Draw diagrams of the typical procedure stack frames. Make sure that you describe the register set, the volatile registers, how parameters and local variables are allocated and accessed.

What are the main advantages of the x64 procedure calling convention compared with that of the IA32?

x64

Briefly describe the IA32 procedure calling conventions. Draw diagrams of the typical procedure stack frames. Make sure that you describe the register set, the volatile registers, how parameters and local variables are allocated and accessed, and the use of the shadow space.

What are the main advantages of the x64 procedure calling convention compared with that of the IA32?

Q2

Virtual Memory

What is the difference between a virtual and a physical address? What is the advantage of using a separate virtual address space for each process?

How are virtual addresses converted to physical addresses by a two level page table? Assume a page table structure as per an IA32 based CPU. Draw appropriate diagrams to illustrate your answer.

What is the advantage of using a two level page table compared with a single level page table? Comment on the amount of physical memory needed for the page tables of (i) a “small” process and (ii) a maximum sized sized process using a single level and a two level page table. Draw appropriate diagrams to illustrate your answer.

Show in detail how the virtual address 0x12345678 is converted into a physical address.

What is demand paged memory management?

What is a TLB? What is a TLB miss? What happens on a TLB miss? Explain (i) how a TLB speeds up the operation of a memory management unit (ii) the effect of a user process switch on the contents of a TLB (iii) how kernel & user virtual addresses can share a TLB and (iv) how TLB misses are handled by a RISC processor.

How are illegal memory accesses detected?

(
A user process is created which has 0x7000 bytes of code, 0x1678 bytes of initialised data, 0x2888 bytes of uninitialised data and 648 bytes of stack data copied from its parent. Draw a diagram that shows the structure and size of the two level page table which is initially created for the process. What size if the initial memory foot print of the processes?
/
A user process is created which has 0x5001 bytes of code, 0x0601 bytes of initialised data, 0x2888 bytes of uninitialised data and 642 bytes of stack data copied from its parent. Draw a diagram that shows the structure and size of the two level page table which is initially created for the process. What size if the initial memory foot print of the process?
)

Pipelining

What is a pipelined processor? What are the benefits of pipelining? Explain the organisation and operation of the DLX five stage execution pipeline.

What are data hazards? Describe two techniques (which prevent stalls) that can be used to overcome data hazards in the DLX pipeline. Show using diagrams, how the data hazards in the following code sequence are overcome by the DLX CPU.

1. r1 <— r2 + r3
2. r4 <— r1 + r1
3. r10 <— r1 + r4
4. r11 <— r4 + r1

What is a load hazard? Describe a technique that can be used to avoid load hazards. Show how the load hazards in the following code sequence can be overcome when executed by the DLX CPU (assume a,b,c,d,e & f are memory locations).

(
1. a <— b + c
2. d <— a - e + f
/
1. r1 <— M[a]
2. r1 <— r1 + r2
3. r2 <— M[b]
4. r3 <— r3 + r3
)

Q3

Cache Basics & Hit/Misses

What is a cache? How does a cache reduce the effective memory access time?

Cache misses can be classified into 3 types - compulsory, capacity & conflict. Explain the meaning of these terms and explain how each could be calculated for a given cache and trace file. What strategy would you advise for reducing each type of miss?

With the aid of diagrams, explain how a cache organisation can be characterised by three constants LKN. Show in detail how a data item is accessed in a LKN cache.

(
Compute the number of hits and misses if the following list of addresses (in hexadecimal) is applied to (a) a 128 byte direct mapped cache with 16 bytes per line and (b) a 128 byte 2-way set associative cache with 16 bytes per line.

0x0000, 0x0040, 0x0008, 0x0040, 0x00C0, 0x0044, 0x0004, 0x0024, 0x0000, 0x00A0, 0x0000, 0x0040, 0x0000, 0x0020, 0x0080, 0x00A0
/
Compute the number of hits and misses if the following list of addresses (in hexadecimal) is applied to (a) a 128 byte direct mapped cache with 16 bytes per line and (b) a 128 byte 4-way set associative cache with 16 bytes per line.

0x0000, 0x0004, 0x0080, 0x0008, 0x0084, 0x0090, 0x0004, 0x008C, 0x0094, 0x0000, 0x0060, 0x0000, 0x0040, 0x0000, 0x0020, 0x0080, 0x0060
/
Compute the number of hits and misses if the following list of addresses (in hexadecimal) is applied to (a) a 128 byte direct mapped cache with 16 bytes per line and (b) a 96 byte 3-way set associative cache with 16 bytes per line.

0x0000, 0x0004, 0x000C, 0x0020, 0x0028, 0x0020, 0x0208, 0x0004, 0x0304, 0x00208, 0x0020, 0x020C, 0x0000, 0x0020, 0x0014, 0x0080
/
Compute the number of hits and misses if the following list of addresses (in hexadecimal) is applied to (a) a 64 byte direct mapped cache with 16 bytes per line and (b) a 64 byte fully associative cache with 16 bytes per line.

0x0000, 0x0010, 0x0020, 0x0030, 0x0034, 0x0020, 0x0010, 0x000C, 0x0050, 0x0040, 0x002C, 0x0008, 0x0030, 0x0020, 0x0010, 0x0000
)

Assume:
1. the low 4 address bits are used as an offset into the cache lien and the next log2(N) address bits select the set.
2. all cache lines are initially invalid
3. an LRU replacement policy

Having computed the misses for the direct mapped and fully associative caches above, determine the number of compulsory, capacity and conflict misses in each case.

Would you expect an associative cache of size N and line size L to always have a higher hit rate than direct mapped cache of size N and line size L? Explain the reasoning behind your answer.

Would you expect a 2-way cache to always outperform a 1-way cache of equal size? Identify an address sequence which generates, for equally sized caches, more misses for the 2-way cache than the 1-way cache(assume a LRU policy). Explain the reasoning behind your address sequence and classify each miss as being compulsory, capacity or conflict. What is your interpretation of a negative conflict miss?

IA32 CPUs often maintain the tags in a set in pseudo least recently used order. Explain how this is achieved using K-1 bits per set. If K = 4 and the tags are accessed in following order 1,2,0,1 & 3, which tag is considered to be the pseudo lease recently used (assume that the K-1 bits are initially 0)?

Q4

Cache Coherency

What is the cache coherency problem? Under what conditions are the caches in a system considered to be coherent?

Clearly define the states used by the Write-Once cache coherency protocol.

Briefly explain the states and operation of the Write-Once and MESI cache coherency protocols.

(
Consider a **three** CPU Multiprocessor system where each CPU has its own cache. Each cache is direct mapped with 2 sets. Even addresses (a0, a2) map to set 0 and odd addresses (a1, a3) map to set 1. The caches are initially empty. Explain in detail the bus traffic and cache line state transitions that occur when the following sequence of “memory” accesses are made by the specified CPU if (1) a Write-Once and (2) a MESI protocol is used.

CPU 0: read a2
CPU 0: write a2
CPU 0: write a2
CPU 1: read a2
CPU 1: read a0
CPU 0: write a2
CPU 0: write a2
/
Consider a **three** CPU Multiprocessor system where each CPU has its own cache. Each cache is direct mapped with 2 sets. Even addresses map to set 0 and odd addresses map to set 1. The caches are initially empty. Explain in detail the bus traffic and cache line state transitions that occur when the following sequence of “memory” accesses are made.

CPU 0: read a0
CPU 0: read a0
CPU 1: read a0
CPU 1: write a0
CPU 2: read a0
CPU 2: write a0
CPU 0: write a0
CPU 0: read a0
CPU 1: read a0
CPU 0: write a0
CPU 1: write a0

Assuming the multiprocessor described above, imagine that each CPU can execute a tight loop which continuously increments a shared variable v. Comment on the rate at which v is incremented if 1,2 or 3 CPUs are concurrently executing the loop. Assume a round robin bus arbiter.
/
Consider a **three** CPU Multiprocessor system where each CPU has its own cache. Each cache is direct mapped with 2 sets. Even addresses map to set 0 and odd addresses map to set 1. The caches are initially empty. Explain in detail the bus traffic and cache line state transitions that occur when the following sequence of “memory” accesses are made.

CPU 0: read a0
CPU 0: read a0
CPU 0: write a0
CPU 0: write a0
CPU 1: read a0
CPU 2: read a0
CPU 2: write a0
CPU 1: write a0
CPU 0: write a2
CPU 0: write a0
CPU 0: write a0

Assuming the multiprocessor described above, imagine that each CPU can execute a tight loop which continuously increments a shared variable v. Comment on the rate at which v is incremented if 1,2 or 3 CPUs are concurrently executing the loop. Assume a round robin bus arbiter.

/
Consider a **three** CPU Multiprocessor system where each CPU has its own local cache. Each cache is direct mapped with 2 sets. Even addresses map to set 0 and odd addresses map to set 1. The caches are initially empty. Explain in detail the bus traffic and cache line state transitions that occur when the following sequence of “memory” accesses are made. Assume a MESI cache coherency protocol. Assume at t=6 and t=7 that ALL memory accesses on each row of the table are complete before initiating the memory accesses on the next row (i.e. a round robin bus arbiter).

t0.	CPU 0: read a0		
t1.	CPU 0: write a2		
t2.	CPU 0: write a2		
t3.		CPU 1: read a2	
t4.		CPU 1: read a2	
t5.	CPU 0: write a2		
t6.	CPU 0: write a2	CPU 1: write a2	CPU 2: write a0
t7.	CPU 0: read a2	CPU 1: read a2	CPU 2: read a0

)

What major advantage does the MESI protocol have over the Write-Once protocol?

What advantage does the write-once protocol have over the simpler write-through scheme if (i) writes are made to an unshared memory location and (ii) writes are made to a shared memory location.