EG 201 Computer Architecture Sep 2019- Midterm
Total - 30 marks, 150 minutes

Instructions:

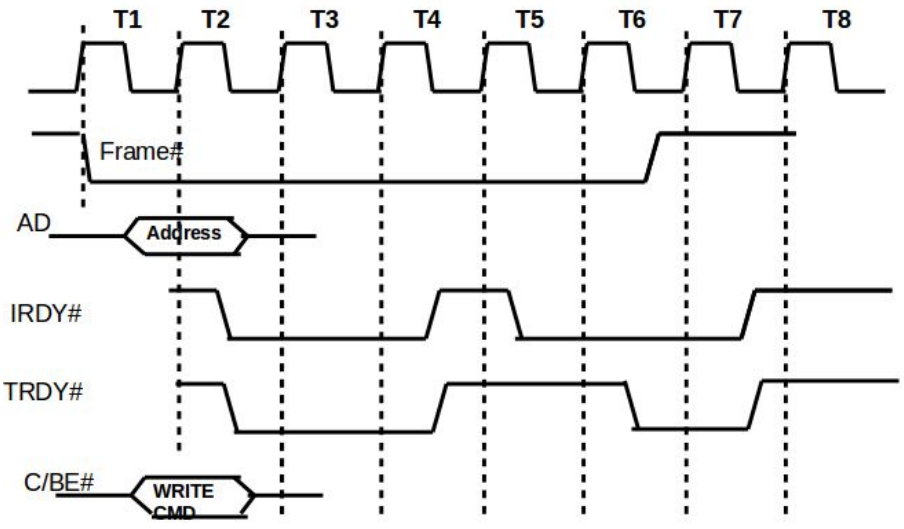
- 1. Write your assumptions wherever applicable
- 2. For multiple choice questions, you need to show how you worked out the solution. Just marking the answer will not fetch any marks
- 3. You should not be in possession of a phone (even if it is off/silent) during the exam. If you have a phone, put it away NOW
- 4. Any malpractice during the exam such as talking, copying etc will fetch a zero for midterm. In addition it will attract a negative 10 marks on overall grading at the end of the course.
- 5. Repeat offenders will get an F grade straight away

Solutions

Partial marks are applicable to all questions.

Ques 1: A PCI Write timing diagram is shown below. Data gets written on the AD bus. Assume positive edge triggered circuit. What are the cycles in which the processor writes the data, but waits for the I/O to read? Why does it have to wait? 3 marks

T5, T6. Either processor is busy (T5) say executing or servicing some interrupt or evaluating data etc or target is busy (T5, T6) - could be because it is still reading in data for eg.



Ques 2:

A processor has an instruction cycle which needs five stages to complete: fetch opcode (three clock cycles), fetch operand address (three cycles), fetch operand (three cycles), execute (say ALU) (three cycles), and store operand (three cycles).
It has a program which executes 3 instructions:
Location 100: Instruction A
Location 101: Instruction B
Location 110: Instruction C

(a) The processor initiates the "fetch operand stage" of the instruction A at the same time that a keyboard activates an interrupt request line. After how long does the processor enter the interrupt processing cycle? What are the sequence of events that follow? **(1+2)**
Interrupt line is received:
Finishes the current instruction A -- that is, the fetch operand (three cycles),execute (three cycles), and store operand (three cycles): 9 cycles

After 9 cycles, stores the context- that is PC (101) on to the stack, and few more registers if needed. Enters the interrupt service routine, returns from the the service routine. Loads the PC content back and continues with Instr B and C.

(b) Let us now introduce 1 cycle of wait in every memory access. After how long does the processor enter the interrupt processing cycle given that the interrupt was introduced at the beginning of fetch operand stage? (Show how you arrived at the answer) **(3)**

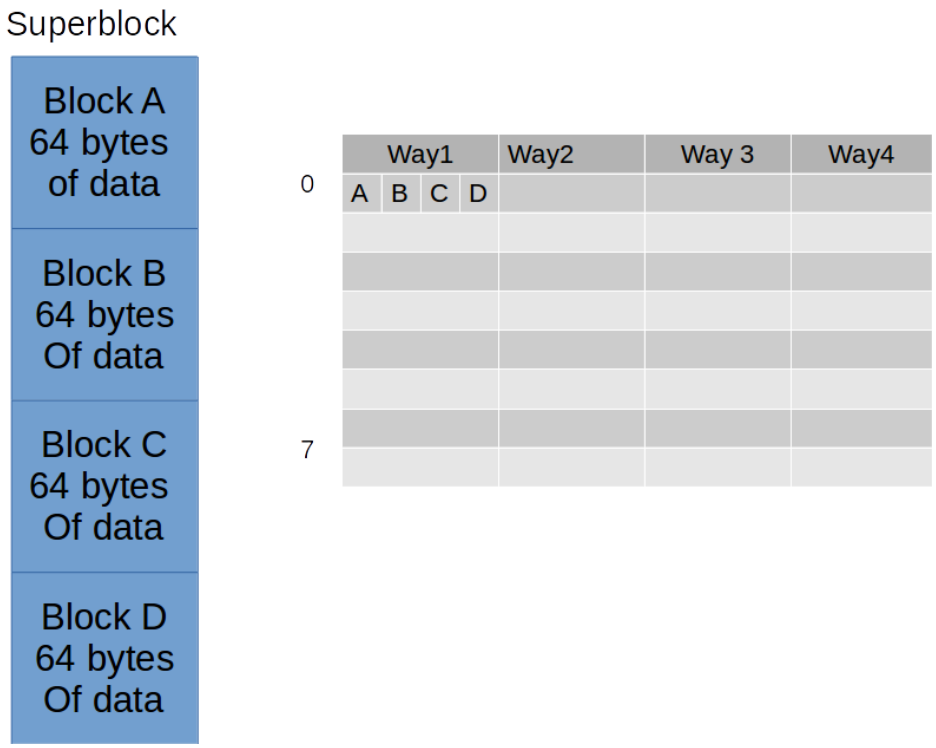
1. **9 cycles**
2. **11 cycles**
3. **10 cycles**
4. **12 cycles**
5. **None of the above**

Soln - Similar to the previous one, but 1 wait are now introduced in memory access. Memory access stages after receiving interrupt are: Fetch operand and store operand, assuming that the operand needs to be fetched from the memory and needs to be stored back to memory
the fetch operand : 3 cycles + 1 cycle wait,
execute- 3 cycles
store operand: 3 cycles + 1 cycle wait,

11 cycles are needed before it can enter the interrupt processing cycle

Ques 3 (Caches)-

Consider an 8 line, 4 way set associative cache as shown in figure below. Each way can store 64 bytes of data per line. The address is 32 bit wide. The data from the memory is fetched in terms of a superblock which consists of 4 blocks of data such as Blocks A, B, C and D for example. Each block contains 64 bytes of data. When the data is fetched, each block is compressed and fit into the cache as shown. So, A. B, C and D are compressed into 16 bytes each and fit into one way of the cache. When the address is a cache hit, the data is decompressed and retrieved.



Solutions:

A. The tag is called the superblock tag. How many bits is the superblock tag? How many bits is the block offset and byte offset? **4 marks**

Data is decompressed into (64 * 4) bytes when fetched from cache. Compression is just to reduce the bytes present in the cache (saves cache space). So, all our definitions of block offset, byte offset, tag etc remain the same.

Soln - 4 blocks → 2 bits of block offset. (1)

64 bytes → 6 bits of byte offset (2)

3 bits of index (8 lines in cache)
21 bits of superblock tag (1)

B. If the processor sends an address 0x000000FF to the cache and it is a cache miss, what is the start and end address of the superblock in memory whose data is fetched into the cache? How many bytes of data is fetched at a time? (**2 marks)**

Soln: Assuming one byte of data per memory location.

0x00FF →corresponds to location 255. Superblock is fetched at a time. Superblock start and end address is 0 to 255. These 256 bytes of data are fetched in → compressed and placed in the cache

Ques 4: (CPI + Memory misses)

Suppose our processor has separate L1 instruction cache and data cache. Our base CPI is 2 clock cycles. Memory accesses take 100 cycles. Our Instruction cache miss rate is 3% while our Data cache miss rate is 10%.

40% of our instructions are loads or stores.

a. What is our processor's new CPI with this kind of a cache? **3 marks**

1. **9**
2. **3**
3. **4**
4. **8**
5. **None of the above**

B. To improve the performance of our processor, we add a unified L2 cache between the L1 caches and memory. Unified L2 cache means that it is a common L2 cache for both instruction and data. An L1 instruction miss or a data miss will go to the unified L2 cache and will then go to the memory. The L2 cache has a hit time of 10 cycles and a miss rate of 2%. What is the new CPI? **3 marks**

1. 5
2. 5.5
3. 4.5
4. 3
5. None of the above

Soln:

Refer to the pdf from UCB I had put up as reading material. The last page of that pdf has these direct formulae

a. 40% (0.4) instructions are load/stores. So, only these instructions access data memory/data cache to load or store data. Rest 60% do not access data cache. All instructions (100% instructions) access L1 instruction cache.

CPI = base CPI + L1 instruction miss cycles + L1 data miss cycles =

2 + 1 * 0.03 MissRate * 100 cycles penalty + 0.4 * 0.1 MissRate * 100 cycles penalty = 2+ 3 + 4 = **9 cycles**

b. Miss rate = 0.02 for L2 - if you consider this as global miss rate (which includes L1 miss rate)
Hit time for L2 is also the miss penalty from L1 to reach L2 = 10 cycles
Assuming same 100 cycle penalty from L2

CPI=CPIbase + L1 instr miss cycles + L1 data miss cycles + L2 instr miss cycles + L2 data miss cycles =
2 + (1 * 0.03 * 10) + (0.4 * 0.1 * 10) + (1 * 0.02 * 100) + (0.4 * 0.02 * 100) = 2 + 0.3 + 0.4 + 2 + 0.8 = 2 + 3.5 = **5.5**
OR

Miss rate = 0.02 for L2 - if you **do not consider** this as global miss rate- then the solution is:
2 + (1 * 0.03 * 10) + (0.4 * 0.1 * 10) + (1 * 0.03 * 0.02 * 100) +
(0.4 * 0.1 * 0.02 * 100) = 2.84
OR
2 + (1 * 0.03 * (10+0.02*100)) + (0.4 * 0.1 * (10+0.02*100)) = 2.84

Ques 5: (3 marks)

(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

Assume a hypothetical computer which has the instruction format as shown here. This machine has the following instruction opcodes:
0001 Load AC from memory
0010 Store AC to memory
0101 Add to AC from memory
0011 = Load AC from I/O device
0111 = Store AC to I/O device
The 12 bit address in the instruction addresses either the memory or the I/O device.

a) Show the instruction codes for this computer for the following program:

1. Load AC from device 5.

2. Add contents of memory location 940.
3. Store AC to device 6
Assume PC starts at 000
Soln -
PC 000: 3005 (assuming device address as 5)
001: 5940 if you assume 940 as hexadecimal. → 53AC if you assume 940 as decimal
002: 7006

b) What is the maximum directly addressable memory capacity (in bytes)? **(2 marks)**

12 address bits → 2^12 = 4k locations each containing one byte→ 4kByte
OR
4k locations * 2 bytes = 8kbytes (2^13) if you assume 16 bits per memory location

Q6 : In a processing system, memory operations currently take 30% of execution time. An "L1 cache" speeds up 80% of memory operations by a factor of 4. A second cache called an "L2 cache" speeds up half of the remaining 20% by a factor or 2. What is the total speed up with L1 and L2? **(Hint- Can Amdahl's law be applied here?) (4 marks)**

a. 2.45
b. 1.237
c. 1.242
d. 0.8
e. None of the above

Soln - Let us Try Amdahl's law:

Just the L1 cache
S1 = 4
b1 = 0.8*0.3 = fraction enhanced
Speedup with L1 = 1/(b1/S1 + (1-b1))
1/(0.8*0.3/4 + (1-(0.8*0.3))) = 1/(0.06 + 0.76) = 1.2195

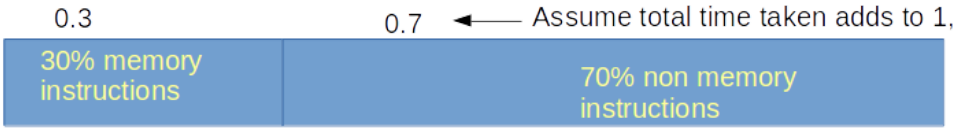
With only L2 cache
S2 = 2
b2 = 0.3*(1 - 0.8)/2 = 0.03
Speedup with L2 = 1/(0.03/2 + (1-0.03)) = 1/(0.015 + 0.97) = 1.015

If you Combine both, speedup = 1.02*1.21 = 1.237

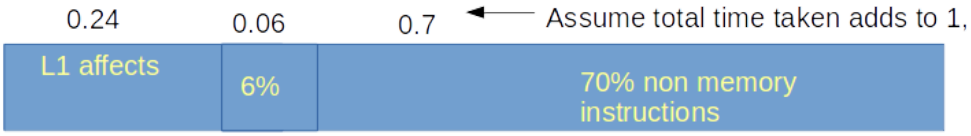
This is not really correct, because after we add the L1 cache, the execution time changes, so the fraction of execution that the L2 affects will change.

Try simple math as follows:

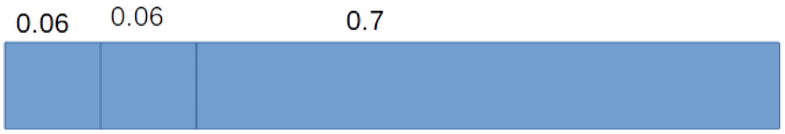
Assume 100 instructions--> 30 are memory, 70 are non memory. Assume 0.1 cycle per instruction. So 0.3 cycles for memory instructions, 0.7 for non-memory instructions



Cache affects only this 30%--> It is given that 80% of these memory instructions are affected by L1 --> which means 0.8 * 0.3 of the time is affected by L1 = 0.24

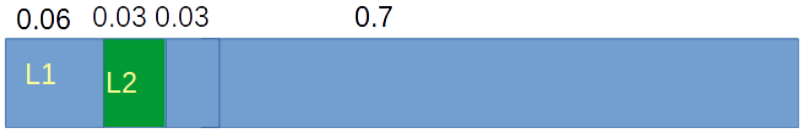


Speed up with L1 is 4--> so memory time changes from 0.24 --> (0.24/4) = 0.06 with L1. New time is 0.06

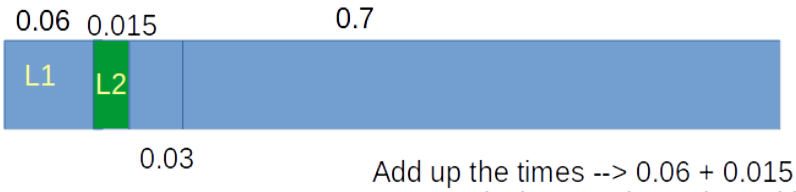


Add up the times --> 0.06 + 0.06 + 0.7= 0.82 is the new time taken with just L1

Now L2 affects half of the remaining 0.06 --> so affects 0.03



L2 improves by a factor of 2 --> 0.03 becomes 0.015



Add up the times --> 0.06 + 0.015 + 0.03+ 0.7
= 0.805 is the new time taken with L1 + L2

Speed up = 1/0.805 = 1.242

If you want to try Amdahl's:
Memory operations = 0.3
SL1 = 4
bL1 = 0.3*0.8 = 0.24
SL2 = 2
bL2 = 0.3*(1 - 0.8)/2 = 0.03

Total speedup = 1/(bL1/SL1 + bL2/SL2 + (1 - bL1 - bL2))
= 1/(0.24/4 + 0.03/2 + (1-.24-0.03)) = 1.24 times

Partial marks are applicable to all questions.