CSCE 513 Computer Architecture
Exam

December 21, 2009

1. Performance - Suppose that the percentage of time that is devoted to floating point adds is 10% of the total execution time and the percentage of time devoted to floating point multiplies is 1% of the total.

   (a) If there are two possible enhancements one to the adder achieves a speedup of 3 and the enhancement to the multiplier achieves a speedup of 20. What is the overall speedup of just the adder?
   `Answer:`

$$\begin{aligned} Speedup_{overall} &= \frac{1}{1 - fraction_{enhanced} \quad + \quad \frac{Fraction_{enhanced}}{Speedup_{enhancement}}} && (1) \\ &= \frac{1}{1 - .1 \quad + \quad \frac{.1}{3}} && (2) \\ &= \frac{1}{.93} && (3) \end{aligned}$$

   (b) What is the speedup of both enhancments?
   `Answer:`

$$\begin{aligned} Speedup_{overall} &= \frac{1}{1 - fraction_{enhanced} \quad + \quad \frac{Fraction_{enhanced}}{Speedup_{enhancement}}} && (4) \\ &= \frac{1}{1 - .1 \quad + \quad \frac{.1}{3}} && (5) \\ &= \frac{1}{.93} && (6) \end{aligned}$$

2. Cache Performance - Suppose in a particular cache that uses write-back, that of the data blocks to be swapped out on average 10% are dirty. Suppose the hit time of the cache is 1 cycle and the miss penalty is 20cycles for the data blocks that are not dirty and 40cycles for those blocks that are dirty. Finally assume the miss rate is 10%.

   (a) What is the speedup up of the memory system if adding a "write-buffer" eliminates 50% of the stall cycles to write back the dirty blocks?
   `Answer:`

$$Speedup_{overall} = \frac{AMAT_{original}}{AMAT_{writeBuffers}}$$

$$\begin{aligned}
AMAT_{orig} &= 1 \;+\; MissRate * MissPenalty \\
AMAT_{orig} &= 1 \;+\; MR * (\%NotD * MP_{Not} \;+\; \%D * MP_D) \\
AMAT_{orig} &= 1 \;+\; .10 \;*\; (.9 * 20 \;+\; .1 * 40) \\
AMAT_{orig} &= 3.2
\end{aligned}$$

$$\begin{aligned}
AMAT_{writeB} &= 1 + MR * (\%NotD * MP_{NotD} + \%D * 50\% * MP_D) \\
AMAT_{writeBuffers} &= 1 \;+\; .10 * (.9 * 20 \;+\; .1 * .5 * 40) \\
AMAT_{writeBuffers} &= 1 \;+\; .10 * (20) \;=\; 3
\end{aligned}$$

$$\begin{aligned}
Speedup_{overall} &= \frac{AMAT_{original}}{AMAT_{writeBuffers}} \\
Speedup_{overall} &= \frac{3.2}{3} \;=\; 1.067
\end{aligned}$$

   (b) What is the maximum possible speedup of the overall memory hierarchy of any write-buffer enhancement for this particluar cache?

3. In the system we are analyzing the memory has:

   - Separate L1 instruction and data caches, HitTime = Processor Cycle Time
   - 32KB L1 instruction cache with 2% miss rate, 64B blocks
   - 256KB L1 data cache with 5% miss rate, 16B blocks
   - 256K L2 unified cache with 64B blocks, local miss rate 20%, Hit Time = 4 cycles,
   - Main Memory Access time is 50 cycles for the first 64 bits and subsequent 64 bit chunks are available every 10 cycles.
   - Both L1 caches are four-way associative, L2 direct mapped.
   - Assume there are no misses to main memory.

   (a) What is the Miss Penalty for accesses to L2?

   `Answer:    64bits=8B transferred at a time, so there is a total of 64B/8B = 8 chunks transferred`

$$\begin{aligned} HitTime_{main-memory} &= initial\ \ chunk\ \ +\ \ 7\ \ remaining\ \ chunks \\ HitTime_{main-memory} &= 50 + 7 * 10\ \ cycles \\ HitTime_{main-memory} &= 120\ \ cycles \end{aligned}$$

   (b) What is the average memory access time for instruction references?

   `Answer:`

$$\begin{aligned} AMAT &= HitTime_{L1} + MR_{L1} * MP_{L1} \\ AMAT &= HitTime + MR_{L1} * (HitTime_{L2} + MR_{L2} * MP_{L2}) \\ AMAT_{Instruction} &= 1 + MR_{L1-Instruction} * (HitTime_{L2} + MR_{L2} * MP_{L2}) \\ AMAT_{Instruction} &= 1 + .02 * (4 + .20 * 120) \\ AMAT_{Instruction} &= 1 + .02 * (4 + 24) \\ AMAT_{Instruction} &= 1.56 \end{aligned}$$

   (c) What is the average memory access time for data references?

   `Answer:`

$$\begin{aligned} AMAT_{Data} &= 1 + MR_{L1-Data} * (HitTime_{L2} + MR_{L2} * MP_{L2}) \\ AMAT_{Data} &= 1 + .05 * (4 + .20 * 120) \\ AMAT_{Data} &= 1 + .05 * (28) \\ AMAT_{Data} &= 2.4 \end{aligned}$$

   (d) Assume the only memory reference instructions are loads(25%) and stores(5%). What percentage of total memory references are data references?

   `Answer:    Let #instructions = I then`
   `#datareferences = .3*I`
   `So the %total memory references that are data references is`

$$\frac{.3*I}{I+.3*I} = \frac{.3}{1.3} = .23$$

(e) What is the Average memory access time?

`Answer:`

$$\begin{aligned} AMAT_{overall} &= \%InstrRef * AMAT_{Instr} \quad + \quad \%DataRef * AMAT_{Data} \\ AMAT_{overall} &= (1 - .23) * AMAT_{Instr} \quad + \quad .23 * AMAT_{Data} \end{aligned}$$

4. Virtual Memory/TLB - Given

- 48 bit virtual addresses
- the page size is 4KB,
- 36 bit physical addresses
- 256KB cache, only one level unified, 16 way associativity, 16B lines
- 64 entry TLB, 8 way associative
- If the virtual address is 0xBCDEF9876548 and if the physical page number is 0x46844 with leading zeroes not shown

(a) What is the page offset field?

Answer:
page size = 4K = $2^2 * 2^{10} = 2^{12}$
pageOffsetFieldSize = $log_2 2^{12}$ = 12 bits = 3 hex digits
PageOffset = 0x548

(b) What is the VPN?

Answer: the rest of the Virt. Addr. = 0xBCDEF9876

(c) How big is a PPN?

Answer: 36 bits - 12 bits = 24 bits

(d) What is the physical address?

Answer: 0x46844548

(e) What is the the cache "block offset" field?

Answer: cache block size = 16B,
so cacheOffsetFieldSize = 4 bits, and cacheOffset = 0x8

(f) What is "set-index" field?

Answer: #lines = cachesize / blockSize = 256K/16 = $2^{18-4} = 2^{14}$
#sets = #lines/associativity = $2^{14}/2^4 = 2^{10}$
sizeOfSetIndex = $log_2 2^{10}$ = 10 bits
setIndex = 10 low bits of PPN = $\cdots$ 0100 0101 0100 = 0001010100 = 0x054

(g) What is the cache "tag" field?

hex	0x	4	6	8	4	4	5	4	8
binary		0100	0110	1000	0100	0100	0101	0100	1000
fields		0100	0110	1000	0100	01 00	0101	0100	1000
fields	01	00 01	10 10	00 01	00 01	00	0101	0100	1000
Field Names	Cache Tag = 0x11A11					Set Index=0x054			CacheOffset=0x8

(h) What happens on a TLB miss?

Answer: The virtual address is looked up in the page table. and the block transferred to the TLB. If not in page table then a page fault occurs.

5. Assume a classic 5-stage pipeline that has been extended to allow mutiple cycle operations, with the execution stage of FP multiplies taking 4 cycles, loads and stores taking 1 cycle, FP adds take 2 cycles and integer operations take 1 cycle. Show how the code below moves through the pipeline assuming that the only forwarding is through the register file. Stop with cycle 17.

   Assume all hazards are detected and handled by inserting stalls to preserve the semantics. The code below was generated for the SimpleScalar and then substantially edited. So there are 32 integer registers: $0, $1, · · ·, $31 and 16 double registers $f0, $f2, · · ·, $f30.

   Show how the code proceeds through the pipeline.

Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
addu	$2,$2,$3																	
l.d	$f0,0($2)																	
mul.d	$f2,$f2,$f0																	
l.d	$f8,1000($2)																	
mul.d	$f4,$f4,$f8																	
add.d	$f6,$f2,$f4																	
s.d	$f6,0($4)																	
subui	$4,$4,#4																	
beq	$2,$4,$L46																	

   Label the first hazard that leads to a stall, classify it, and then describe how it could be best eliminated.

Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	1
addu	$2,$2,$3	F	D	E	M	W														
l.d	$f0,0($2)		F	.	.	D	E	M	W											
mul.d	$f2,$f2,$f0					F	.	.	D	E	.	.	.	M	W					
l.d	$f8,1000($2)								F	D	.	.	.	E	M	W				
mul.d	$f4,$f4,$f8									F	.	.	.			D	E	.	.	.
add.d	$f6,$f2,$f4															F	D	.	.	.
s.d	$f6,0($4)																F	.	.	.
subui	$4,$4,#4																			
beq	$2,$4,$L46																			

The first hazard is a RAW hazard in that $2 is calculated by the addu and used in the following l.d. It can eliminated by forwarding from the output of the ALU to the input of the ALU in the next cycle.

6. For the same loop and same assumptions on execution module times and assuming a branch delay slot and the best possible forwarding, schedule the code (rearrange the code making slight changes to preserve semantics) to avoid as many stalls as possible. Tell how many cycles a loop requires.

7. For the same loop and same assumptions on execution module times show how Tomasulo's Algorithm would schedule the operations. For the "Issue," "Execute" and "Write CDB" columns enter the cycle number in which the event is done as opposed to a check mark which was done in the text.

   In particular show the tables when the Add.d receives its operands and begins execution.

Instruction		**Iteration**	Issue	Execute	Write CDB
L.D $f0, 0($2)		1			
MUL.D $f8, $f2, $f0		1			
L.D $f8, 1000($2)		1			
MUL.D $f4, $f4, $f8		1			
ADD.D $f6, $f2, $f4		1			
S.D $f6, 0($4)		1			

Reservation Stations

Name	**Busy**	**Op**	**Vj**	**Vk**	**Qj**	**Qk**	**A**
Load 1							
Load 2							
Add 1							
Add 2							
Mult 1							
Mult 2							
Store 1							
Store 2							

Register Status

Field	F0	F2	F4	F6	F8	F10	F12	F···	F30
Qi									

Answer:

8. **ANSWER** For the same loop and same assumptions on execution module times show how Tomasulo's Algorithm would schedule the operations. In particular show the tables when the Add.d receives its operands and begins execution.

Instruction		**Iteration**	Issue	Execute	Write CDB
L.D $f0, 0($2)		1	1	2	3
MUL.D $f8, $f2, $f0		1	2	7	8
L.D $f8, 1000($2)		1	3	4	5
MUL.D $f4, $f4, $f8		1	4	9	10
ADD.D $f6, $f2, $f4		1	5		
S.D $f6, 0($4)		1	6		

Reservation Stations

Name	**Busy**	**Op**	**Vj**	**Vk**	**Qj**	**Qk**	**A**
Load 1							
Load 2							
Add 1	Yes	ADD.D	val(Mult1)	val(Mult2)			
Add 2							
Mult 1							
Mult 2							
Store 1	Yes	Store	Add 1(F6)				0($4)
Store 2							

Register Status

Field	F0	F2	F4	F6	F8	F10	F12	F···	F30
Qi									

9. Loop Unrolling - Given the code

```
$L46:
        addu    $2, $2, $3
        l.d     $f0, 0($2)
        mul.d   $f2, $f0, $f0
        add.d   $f4, $f2, $f0
        s.d     $f4, 0($4)
        subui   $4, $4, #8
        beq     $2, $4, $L46
```

(a) Loop Unrolling SimpleScalar - Show how to unroll the loop just once.

```
$L46:
        addu    $2,   $2,   $3
        l.d     $f0,  0($2)
        l.d     $f6,  8($2)
        mul.d   $f2,  $f0,  $f0
        mul.d   $f8,  $f6,  $f6
        add.d   $f4,  $f2,  $f0
        add.d   $f10, $f8,  $f6
        s.d     $f4,  0($4)
        s.d     $f10, 0($4)
        subui   $4,   $4,   #16
        beq     $2,   $4,   $L46
```

(b) If your loop would normally execute an odd number of times and your unroll it once, what problems arise and how do you address them.

`Answer:    Handle the work done normally in the last iteration outside the loop.`

(c) What in the architecture limits the number of times a loop could be unrolled?

`Answer:    The number of registers.`

(d) Show how to unroll this loop for a SuperScalar (VLIW) that can issue one integer, one floating add, one floating multiply, and one load/store operation each cycle.

`Answer:`

Cycle	Integer	FPadd	FPmult	Load/Store
1				
2				
3				
4				
5				

10. Describe in detail the steps in the process of translation from a virtual address to a physical address.

    Answer:

    (a) Take the virtual address(VA) and partition it into the virtual page number (VPN) and the page offset (PO).

    (b) Take the VPN and partition into index and tag depending on the associativity of the Translation Lookaside Buffer (TLB).

    (c) Use the index to access the appropriate set in the TLB.

    (d) Compare the tag in VA with the tag(s) in the set.

    (e) i. If equal then contruct the Physical address (PA) by concatenating the Page Offset onto the Physical Page Number (PPN), which is the data in the TLB.

        ii. if the tag is not equal then we have a TLB miss and in this case you go to the page table.

        iii. If the VPN entry is in the page table then we construct the Physical Address by concatenating PPN and Page Offset.

        iv. If page table entry corresponding to this VPN is not valid, then there is a page fault.

Given the code below and a direct mapped cache with only 8 lines.

11.
```
double a[64][64];
double b[64];
double c[64];
double sum = 0.0;
for(i=0; i < 64; ++i)
   for(j=0; j < 64; ++j)
      c[i] = a[i][j] * b[j] + c[i];
```

(a) Assuming that &a[0][0] maps to block 0, &b[0] maps to block 2 and &c[0] maps to block 4, what is the Hit ratio if we assume the blocks are 32B?

(b) What is the hit ration if the loop order is changed so that the "j-loop" is the outer loop?

(c) What is the hit ratio if we change the above scenario by mapping &c[0] to block 2 as well as &b[0].

Answer:

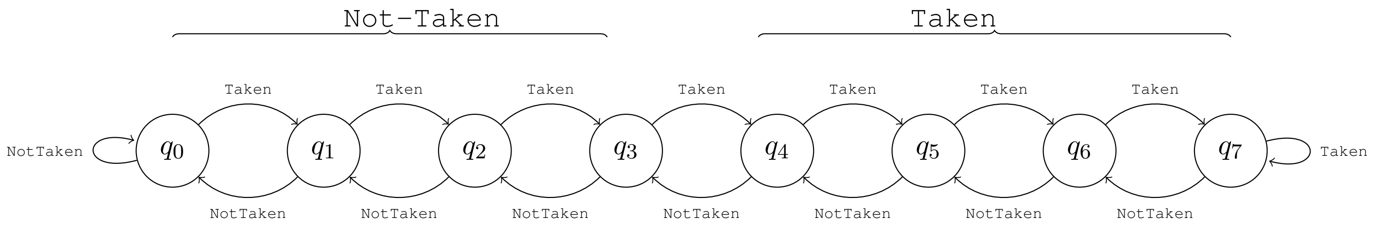
ANSWER

Sets	**A**	**B**	**C**
0	a[i][0] ··· a[i] [3]	b[24] ··· b[27]	c[16] ··· c[19]
1	a[i][4] ··· a[i][7]	b[28] ··· b[31]	c[20] ··· c[23]
2	a[i][8] ··· a[i][11]	b[0] ··· b[3]	c[24] ··· c[27]
3	a[i][12]··· a[i][15]	b[4] ··· b[7]	c[28] ··· c[31]
4	a[i][16]··· a[i][19]	b[8] ··· b[11]	c[0] ··· c[3]
5	a[i][20]··· a[i][23]	b[12] ··· b[15]	c[4] ··· c[7]
6	a[i][24]··· a[i][27]	b[16] ··· b[19]	c[8] ··· c[11]
7	a[i][28]··· a[i][31]	b[20] ··· b[23]	c[12] ··· c[15]

12. Branch Prediction

(a) Draw the diagram for a 3-bit saturating counter branch predictor.

`Answer:`



(b) What is the miss prediction rate of the "Branch-Taken" predictor with the following code. Ignore all branches except the one based on the evaluation of "i % 5"

```
for(i=0; i <1000; ++i){         loop:
  ...
   if (i % 5 == 0) {                 jeq    $then-code
        code segment A               .
   }else{                            .
        code segment B               .
   }
}
```

```
Answer:    Branch-Taken always predicts that we take
the branch.
Consider the sequence of values of i and whether the
branch is taken(T) or not(N)
TNNNNTNNNNTNNNN...
So the percentage mis-predicted is 4/5 = 80%
```

(c) What is the miss prediction rate of a 2 bit saturating counter predictor with the same code. Again ignore all branches except the one based on the evaluation of "i % 5"

```
Answer:    Now consider any sequence of 5 consecutive
values of i, there will be 4 branches not-taken and
one taken.  So regardless of what the initial state is
quickly the state machine will move to state q0 and it  and
will stay there or close and it will always after that
predict ``Not-Taken.''
So miss-prediction rate = 20% (or real close)
```

(d) Describe how a (m,n) correlating branch predictor works.

`Answer:    The predictor using the last m branches to select one of` 2^m `n-bit saturating counters to predict the branch.`

13. Given a four core system (4CPUs on a chip) with each CPU having a separate L1 cache.
    (a) Give an example sequence of operations that shows the need for Snoopy cache
    (b) Explain the difference in fine and coarse grained threading.