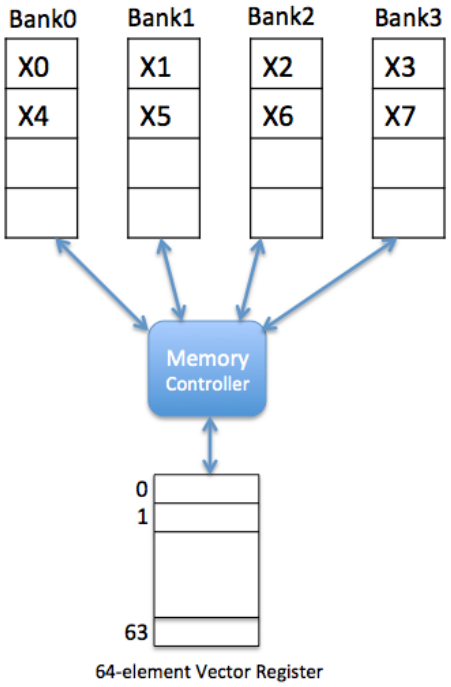
Computer Science 605.411

Final Example Set 1

1. The diagram below represents an interleaved memory system containing four 32-bit wide banks or modules that can operate in parallel. The memory controller connects the memory banks to a vector register over a 32-bit bus and sequences accesses to the banks. If the time to access each bank is 4 clock cycles, what is the total number of clock cycles required to completely fill the 64-element vector register with the 64 elements of the memory resident vector X? To save space, only the first 8 elements of X are shown.



In the first cycle, access to X0 in bank 0 can be started and will require 4 cycles to complete. A separate access to each of the remaining banks (1 through 3) can be started in each of the next 3 clock cycles. At the end of the 4th cycle X0 will be available and access to X4 can be started. Hence, it will take 4 clock cycles to fill the first element in the vector register, and each subsequent element will require one additional clock cycle to fill. Therefore completely filling the vector register will take 4 + 63 = 67 clock cycles.

2. What happens if consecutive vector elements are not in adjacent memory modules?

In this case, instructions such as LVWS (load vector with stride) and SVWS (store vector with stride) would be used to access the vector elements in memory. The distance or displacement in bytes between consecutive vector elements would be placed into a separate scalar register and used in generating the address of each vector element. For example, if consecutive integer elements appear in every other memory bank, the stride would be 2 and the value placed into the stride register would be 2*4 = 8. The following code would load a 5-element vector (VX) with stride 2 into the vector register $V0:

```
ori     $9,$0,8             #place stride displacement into $9
la      $8,VX               # put address of vector Vx into $8
ori     $7,$0,50            # set length register to reference 50 elements
LVWS  $V0,($8),$7,$9    # read vector VX into vector register $V0
```

3. Suppose that a vector processor that employs the memory system described above, has 64 adders each of which requires 1 clock cycle to produce the sum of its two inputs. Considering only the time required to obtain the operands from memory, compute the vector sum, and store the vector result back into memory, what is the total number of clock cycles required to carryout the following high level vector instruction without chaining?

C = A + B # A, B and C are 64-element integer vectors.

Without chaining, both vectors would be loaded into a separate vector register, then the vector sum would be generated, followed by storing the contents of the result vector register into memory. It takes 67 cycles to load one register with vector A and 67 cycles to load vector B into another vector register. Then the 64-element sum would be generated in one cycle by the 64 adders operating in parallel. So it would take 2*67 + 1 = 135 cycles to generate the vector sum. It would then take 67 cycles to store the vector result into memory.

4. How would vectors whose lengths are not the same as the length of the vector registers be handled?

The length register would be set to the size of the vectors that are less than the length of the vector register. If the length of the vector (say N) is greater than the number of elements in the vector register, then it would partitioned into multiple groups of N elements, and the final group would contain any left over elements.

5. An integer scalar value can be loaded into register $5 using the following instruction: lw $5,0($4) where $4 contains the memory address of the scalar value. How does this instruction differ from the following vector load instruction:

LDV $V1, 0($4),$8

where $V1 is a 64-element vector register and $4 contains the memory address of a 64-element integer vector and $8 is the length register indicating the number of elements in the vector?

The scalar load instruction would generate just one memory address and cause the single word at that address to be read from memory and loaded into the scalar register $5. The vector load instruction would generate a series of memory addresses (corresponding to the locations of the vector elements in memory) and would perform a read from each address, with each value read going into the next element within the vector register $V1.

6. How could chaining be used to generate the vector sum C = A + B using just one adder?

Assuming that the elements of the vectors A and B are in different memory modules, the reading of elements from A could be interleaved with reading elements from B (i.e., A0, B0, A1, B1, etc). As soon as the first elements of the vector registers for A and B are filled, the A and B elements could be added together. Each addition would be done in parallel with the reading of the next vector elements from memory.
The following table shows what happens in the first seven cycles:

Cycle	Operations
1	Read A0
2	Read B0
3	Read A1
4	Read B1, A0 available
5	Read A2, B0 available compute C0=A0+B0
6	Read A3, A1 available
7	Read B3, B1 available, compute C1

Hence the first element in the sum is available after 5 cycles, with each subsequent element becoming available in an additional 2 cycles per element for a total of 5 + 2*63 = 131 cycles which is faster and uses 63 fewer adders.

7. A uniprocessor has an L1 and an L2 cache. With no cache misses, the processor achieves a CPI of 1. Suppose that in reality an L1 miss occurs every 50 cycles and that an L2 miss occurs every 500 cycles. The L1 miss penalty is 40 cycles and the L2 miss penalty is 400 cycles. What would be the effective (i.e., average) CPI rating for the processor with these cache miss rates?

The L1 miss rate of 1 every 50 cycles corresponds to one L1 miss for every 50 instructions. The L2 miss rate corresponds to one L2 miss for every 500 instructions. Hence every 50 instructions would take 50 + 40 = 90 cycles; every 500 instructions would take 10*90 + 400 = 1300 cycles. Therefore the average CPI = 1300/500 = 2.6.

8. Three threads A, B and C are executed on a fine-grained multi-threaded version of our MIPS 5-stage pipeline system with no data hazard unit and no forwarding unit and no branch prediction. Thread A consists for 5 instructions (A0, A1, A2, A3 and A4); the instructions in thread B are B0 through B4 and those in thread C are C0 through C4. Show how the three threads would flow through the pipeline assuming no cache misses.

Since the system employs fine-grained multi-threading, the instructions from the three threads would be interleaved in a round-robin fashion within the pipeline. For each clock cycle, an instruction from a different thread would enter the pipeline until all instructions have been executed:

Cycle	Fetch	Decode	Execute	Memory	Write-back
1	A0				
2	B0	A0			
3	C0	B0	A0		
4	A1	C0	B0	A0	
5	B1	A1	C0	B0	A0
6	C1	B1	A1	C0	B0
7	A2	C1	B1	A1	C0
8	B2	A2	C1	B1	A1
9	C2	B2	A2	C1	B1
10	A3	C2	B2	A2	C1
11	B3	A3	C2	B2	A2
12	C3	B3	A3	C2	B2
13	A4	C3	B3	A3	C2
14	B4	A4	C3	B3	A3
15	C4	B4	A4	C3	B3
16		C4	B4	A4	C3
17			C4	B4	A4
18				C4	B4
19					C4

b) If instruction A1 requires the result from instruction A0, how many pipeline bubbles would result?
No bubbles would be required since by the time A1 reaches the decode stage, A0 will have written the result register in the write-back stage.

c) If instruction A2 is a conditional branch instruction that is taken, how many pipeline stages would have to be flushed?
The conditional evaluated by A2 determines which instruction from thread A should be executed after A2. No stages would have to be flushed since the condition tested by A2 would be complete when A2 finishes the execute stage, so the instruction within thread A to which A2 branches can be brought into the pipeline in the next cycle.

Computer Science 605.411

Final Example Set 2

1. Consider a quad-core system in which each core has a separate L1 cache with a miss penalty of 20 cycles. The cores share a common L2 cache with a miss penalty of 200 cycles. Suppose that one of the cores experiences an L1 miss ratio of 5% and an L2 miss ratio of 0.5%. The core has a CPI of 1 in the absence of cache misses.

a) What is the average CPI of the core when these cache miss rates are taken into account?
The average number of cycles per instruction would be 1 plus the penalty per instruction due to the cache misses.
So the average CPI = 1 + 0.05*(20 + 0.005*200) = 1 + 1.05 = 2.05

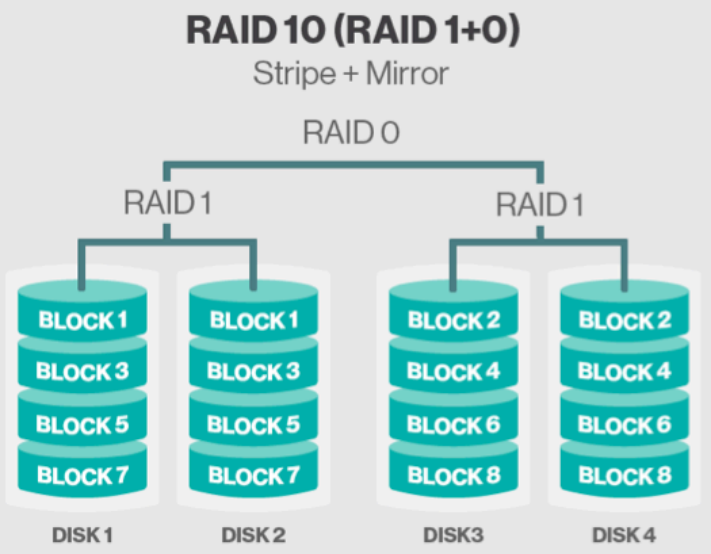
b) Suppose that miss occurs in the L1 cache every 100 cycles, how many cycles would this miss L1 rate add to the base CPI of 1?
For a base CPI of 1, an L1 miss every 100 cycles would be a miss every 100 instructions. Hence every 100 instructions would take 100 + 20 = 120 cycles which corresponds to an extra 0.2 cycles per instruction.

c) Suppose that a miss occurs in the L2 cache every 400 cycles, how many cycles would this L2 miss rate add to the base CPI of 1?
For a base CPI of 1, an L2 miss every 400 cycles would be a miss every 400 instructions. Hence for every 400 instructions the L2 miss rate would add an extra 200 cycles or 0.5 cycles per instruction. Recall that only the references that miss in the L1 cache are sent to the L2 cache.

2. How many bits would be required for each set in an 8-way set associative cache to implement a pseudo-LRU replacement algorithm?
The eight ways in a set can be viewed as a pair of 4-way groups. We know that a 3-bit pseudo-LRU scheme can be used for each group of 4 ways. A separate bit can be used to indicate which of the two 4-way groups is least recently used. Hence a total of 7 pseudo-LRU bits are required.

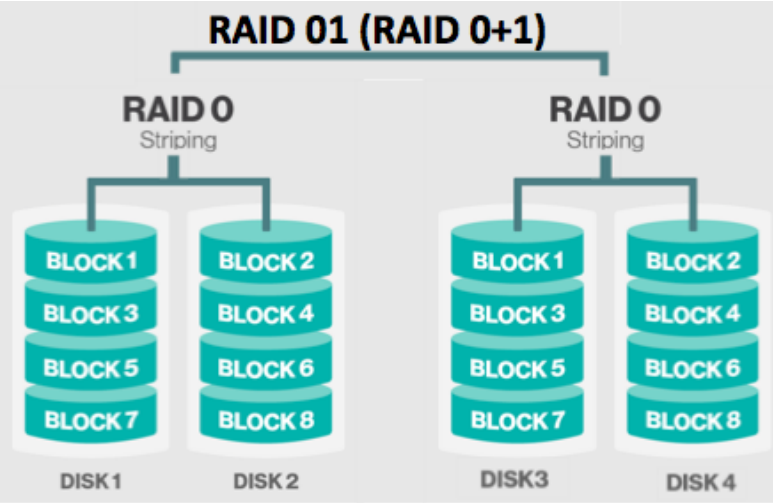
3. For processors that require one or more cycles to execute each instruction, the performance can be defined in terms of the average number of cycles per instruction (CPI). However, for processors that execute more than one instruction per cycle, a better definition of performance would be based on the average number of instructions completed per cycle (IPC) . Write down an expression for performance as a function of IPC, clock rate(CR) and instruction count (IC).
Performance = ___(IPC*CR) / IC ___
By definition performance = 1/execution_time = 1/(IC*CPI*cycle_time)
= 1/(IC * CPI * 1/CR) = 1/ (IC * 1/IPC * 1/CR) = 1/(IC/(IPC*CR))
= (IPC*CR) / IC

4. Suppose that each of the 4 processors in an SMP (shared memory multi-processor) system is rated at 200 MIPS, what would be the apparent MIPS rating for a program that can be partitioned into four independent parts (A, B, C and D) where part A accounts for 35% of the instructions in the program, part B accounts for 48%, part C accounts for 12%, and Part D accounts for 5%?
The four parts would be executed in parallel. The time required to execute the program would correspond to the longest part (part A), which is T= 0.48*IC/200 MIPS. So the apparent MIPS rating would be IC/T = 200 MIPS/0.48 = 416.67 MIPS.5. a) How many stripe controllers and how many mirror controllers are required for the RAID 10 system shown below?



Two mirror controllers are required (one for each of the two mirrored pairs) and one stripe controller is required to manage the RAID 0 striping.

b) How many stripe controllers and how many mirror controllers are required for the RAID 01 system shown below?



Two stripe controllers are required (one for each of the two RAID 0 systems) and one mirror controller for the RAID 0 mirror images.

c) How many disks would the operating system see with the RAID10 and RAID01 systems shown above?

Although each strip controller sees two disks and each mirror controller sees two disks, each system will appear as a single disk to the operating system.

Computer Science 605.411

Final Example Set 3

1. Which one of the following is the main reason for employing delayed branches?
    a) to prevent the branch from executing too quickly
    b) to avoid data dependencies
    c) to reduce pipeline bubbles
    d) to aid in speculative execution
    e) to help predict branch behavior

The main reason is to reduce the bubbles that result from having to flush the pipeline when a branch is taken. Even if the branch is predicted in stage 2 or if the branch condition is evaluated in stage 2, the next instruction following the branch will already be in the fetch stage. Delayed branching eliminates the bubble that would result if this instruction is flushed.

2. A certain byte-addressable computer system employs 13 bit virtual addresses but only contains four 1024-byte frames. Shown below is the contents of the page map table at the time that a reference to decimal address 6494 is made.

Page Table

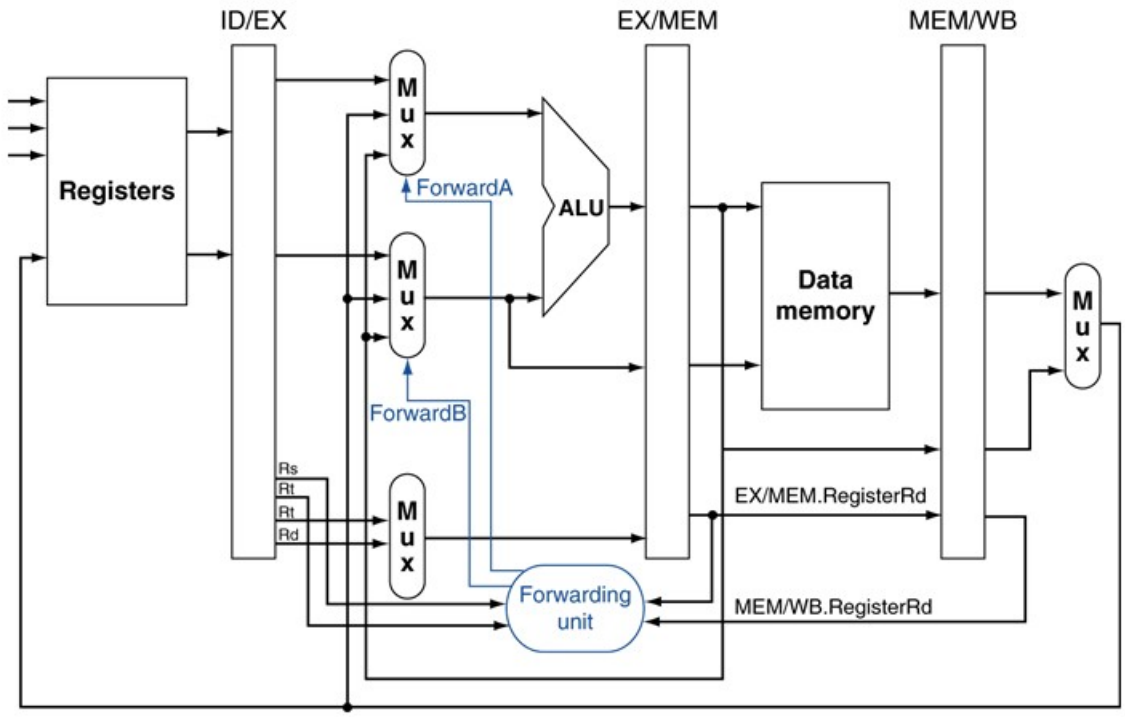
Page	Frame	Valid Bit
0	–	0
1	3	1
2	0	1
3	–	0
4	–	0
5	1	1
6	2	1
7	–	0

Would this reference cause a page fault? If not, what is the corresponding physical address?

Address 6494 = 0x195E = 1100101011110 which maps to page 6. The valid bit in PTE 6 is 1, so there is no page fault. The corresponding frame number is 2. Therefore the physical address = 100101011110 = 0x95E = 2398.

3. The diagram below shows the instructions that occupy the 5 stages during a certain clock cycle within our MIPS pipelined system that includes a forwarding unit:

Fetch	Decode	Execute	Memory	Write-back
Beq $2,skip	Slt $2,$4,$7	??????	Add $7,$4,$5	Lw $9,4($12)

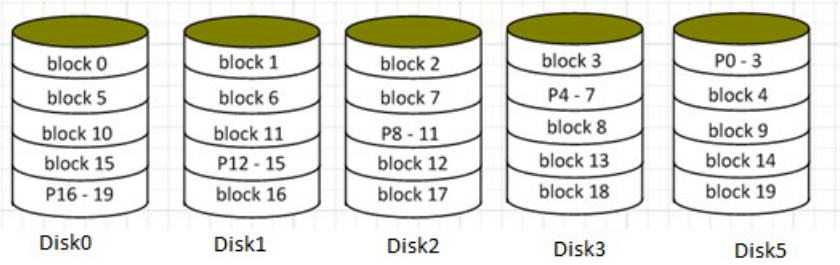


Which one of the following instructions could not be allowed to occupy the execute stage during this clock cycle?

a) add $7,$7,$9
b) beq $7,$9,exit
c) lw $8,0($9)
d) sw $7,4($9)

The answer is d) sw, since it does not use $7 as an ALU input, and the datapath as shown only allows forwarding to the ALU inputs. The required lower ALU input is the sign extended constant 4. The upper ALU input is $9.

4. Consider the following RAID5 system that uses a simple bit-wise XOR parity scheme:



Suppose that disk2 fails and is replaced by a new blank disk which is to be written with the information that had been on the original disk2.

a) Write down an expression for the parity block and for each of the four data blocks that should be written to the new disk2.

P8-11 = _____block10 XOR block11 XOR block8 XOR block9_____

Block2 = ____block0 XOR block1 XOR block3 XOR P0-3___ Block7

= ____ block5 XOR block6 XOR block4 XOR P4-7_____

Block12 = ___ block15 XOR block14 XOR block13 XOR P12-15_____

Block17 = ___ block16 XOR block18 XOR block19 XOR P16-19___

b) Assume that a disk read or write operation takes T time units and that the time to compute the XOR of two or more strips is also T time units. What is the minimum total number of time units required to reconstruct the complete new disk2 image? That is, if the minimum total time required to reconstruct disk2 is N*T, what value is N? N = _

Blocks on separate disks can be read in parallel (T units since each block is read from a different disk), the XOR takes T units (they are done in parallel), then the reconstructed block is written to disk2 (T units). Therefore a minimum of 3T units are required to reconstruct and write each of the missing blocks. Hence a total of 5*3T = 15T is required. So N=15.

5. A computer system includes a single unified primary cache that contains L lines. The total number of memory blocks in the central memory is M (where M and L are both powers of 2). If S is the total number of sets in the cache, how many different blocks from central memory would be a candidate to occupy the very first line within a single set for each of the following cache organizations:

a) Direct mapped number of candidates = _ M/L_

The total number of memory blocks M divided by the number of cache lines.

b) Four---way set associative number of candidates = _ M/S = M/(L/4) = 4M/L_

This is the total number of blocks divided by the number of sets S=L/4.

c) Fully associative number of candidates = _ M_(i.e. any block can reside in any line)___

6. Suppose that only 1% of the references made by a certain program result in page faults.

It takes 200ns to complete each access if no page fault occurs. The total time required to handle each page fault and resume the program is 10 milli-seconds.

a) What would be the effective (i.e. average) time per access for the program?

Each reference must first check the PMT (200ns)
If there is no page fault, the operand is accessed (200ns)
10 milli-seconds = $10^* 10^{-3}$ seconds = 10^{-2} = $10^7 * 10^{-9}$ = 10000000 ns.
Each page fault takes 10000000 ns (10 ms) to handle. Once the page is loaded into memory and the PMT updated, the instruction that caused the page fault is re-executed.
The effective access time =0.99 * (200 + 200) + 0.01* (200 + 10000000 +200 + 200)
= 396 + 100006ns = 100402 ns

b) What would be the effective (i.e. average) time per access for the program if no page faults occurred?

The effective access time =1.0 * (200 + 200) = 400 ns (the page table must be accessed for every reference, then the operand is accessed)

7. A version of our 5-stage pipelined system employs fine-grained multi-threading to run three threads by executing one instruction from each thread per cycle on a round-robin basis. That is, one instruction from thread A, followed by one from B then one from C.

a) Show how the instructions would flow through the pipeline.

Cycle	Fetch	Decode	Execute	Memory	Write-back
1	A1				
2	B1	A1			
3	C1	B1	A1		
4	A2	C1	B1	A1	
5	B2	A2	C1	B1	A1
6	C2	B2	A2	C1	B1
7	A3	C2	B2	A2	C1
8	B3	A3	C2	B2	A2
9	C3	B3	A3	C2	B2
10	A4	C3	B3	A3	C2
11		A4	C3	B3	A3
12			A4	C3	B3
13				A4	C3
14					A4

b) If the third instruction from thread A (A3) depends on a result from the second instruction of thread A (A2), how many bubbles would result assuming that ONLY a data hazard unit is employed?

No bubbles would be required since A2 would be in the write-back stage when A3 is in the decode stage, so A3 would read the updated registers.

c) How many bubbles would result if no branch prediction is used and the second instruction from thread B (B2) is a conditional branch that is taken?

The instruction B2 would take effect when it is in the memory stage (stage 4) and would set the PC for thread B to fetch the appropriate instruction. No other instruction from thread B would be in the pipeline, so no flushing would be required and no bubbles would be produced.

8. A virtual memory system has a page fault frequency of 2%. Twenty cycles are required to access the main memory and it takes 4000 cycles to handle a page fault, load a page from disk and resume the program. The system also includes a fully associative TLB with an 5 cycle access time and a hit ratio of 80% as well as a unified 4-way set associative cache that provides an average access time of 10 cycles per access. If the CPU runs at a 2GHz clock rate, how much does using the TLB reduced the effective access time compared to not using the TLB? Express your answer in nano-seconds.

The 2 GHz clock rate corresponds to a 0.5 ns cycle time.
The average access time includes the time to obtain the physical address plus the time required to access the item at that physical address.

Without the TLB, every access would have to check the PMT.

References to items in memory take 20 + 20 = 40 cycles

Reference to items not in memory take 20 + 4000 + 20 + 20 = 4060 cycles

So the average access time = 0.98*40 + 0.02*4060 = 39.2 + 81.2 = 120.4 cycles or 60.2 ns

With the TLB, the average time to obtain the physical address = 0.8*5 + 0.2(5+20) = 9 cycles.

References to items in memory take 9 + 20 = 29 cycles on average.

Reference to items not in memory take 9 + 4000 + 9 + 20 = 4038 cycles

So the average access time = 0.98*29 + 0.02*4038 = 28.42 + 80.76 = 109.18 cycles

or 54.59 ns

9. Consider a system with a Harvard Architecture. Which one of the following would be the best valid reason for using low order interleaving for the separate instruction memory?

a) to speedup access to instruction memory operands
b) to reduce the time that the CPU must wait for instructions
c) to avoid unaligned memory access exceptions
d) to facilitate the use of little endian storage order

The instruction memory only contains instructions, no memory operands.
Low order interleaving would allow multiple modules in the instruction memory to be read in parallel, thus speeding the access to instructions. This would reduce the time that the CPU has to wait for instructions. So the correct choice is b)

10. Assume that the following instruction is executed:

```
Lui   $14,0x73E4
Ori   $14,$14,0x82CA
Lui    $8,0x1001
Sw    $14,4($8)
Lb    $9,6($8)
```

Use 8 hex digits to show the pattern left in register $9 by this instruction sequence assuming:

a) (5) little endian memory storage order is used $9 =____0xFFFFFFE4_____

The first two instructions place 0x73E482CA into $14.
The lui instruction places 0x10010000 into $8
The sw then stores the contents of $14 into the 4 bytes starting at address 0x10010004 in the order CA,82,E4,73 so the byte at address 0x1001006 contains 0xE4 which is loaded into the low byte of $9 and sign-extended through out the remaining bytes in $9.

b) (5) big endian memory storage order is used $9 =_____0xFFFFFF82___

The bytes are stored in the order 0x73,0xE4,0x82, 0xCA with big endian order.

So the byte at address 0x1001006 contains 0x82 which is loaded into the low byte of $9 and sign-extended through out the remaining bytes in $9.

Computer Science 605.411

Final Example Set 4

1. How could performing memory-mapped I/O on an IA-32 processor cause data to be missed when used in conjunction with a data cache?

If the memory block that contains the device status and data registers is brought into the data cache, thereafter any references to those registers would obtain possibly stale information from the cache rather than current data from the actual device register. Accesses to the cache made by I/O devices do not go through the cache, so the CPU may read information from cache that falsely indicates the current state of the I/O device.

2. Which of the following processors employs port-mapped I/O?
a) SparcV8
b) ARM
c) IA-32
d) MIPS

3. How does the ARM exception vector table differ from the trap vector table used on the SparcV8.
The ARM exception vector table begins at address 0 and contains 7 entries (one for each of the seven exception types). Each 4-byte entry contains an instruction that transfers control to a routine designed to handle the particular type of exception.
The SparcV8 trap table is located in memory at the address indicated by the contents of the trap base register (TBR). The trap type (tt) field within the TBR indicates the nature of the trap and identifies one of 256 entries, each of which is four words in size and contains the first 4 instructions of the corresponding trap handling routine.

4. How does the trap vector table used on the SparcV8 differ from the interrupt vector table used on an IA-32 processor?
The IA-32 interrupt vector table typically resides at address 0 in RAM and contains 256 entries each of which is 4 bytes in size. Each entry contains the address of the routine designed to handle the particular type of interrupt.
The SparcV8 trap table is located in memory at the address indicated by the contents of the trap base register (TBR). The trap type (tt) field within the TBR indicates the nature of the trap and identifies one of 256 entries, each of which is four words in size and contains the first 4 instructions of the corresponding trap handling routine.

5. What is the purpose of the LDMIA instruction on the ARM processor?
This instruction reads multiple words from memory into a specified list of registers. A base register contains the beginning address of the block of memory words and is incremented by 4 after each word is transferred.

6. What is the LOOP instruction on an IA-32 processor?
This instruction decrements the count register, tests the result and branches to the indicated destination if the count register does not contain zero.

7. Can an IA-32 processor only perform port-mapped I/O?
IA-32 processors can perform memory-mapped as well as port-mapped I/O.

8. What is the purpose of the "save" instruction on a SparcV8 processor?
The save instruction causes the register window to slide to reveal a new group of 24 registers. The first 8 registers (the 'in' registers i0 through i7) overlap with the 'out' registers o0 through o7 of the previous window. The next 8 register in the window are new local registers l0 through l7. The third group of 8 registers are the out registers (o0 through o7) for the current window.
The same 8 global registers are shared by all register windows. The save instruction can also allocate stack space by decrementing the stack pointer by an specified amount.

9. What value will be in the PC when an ARM instruction whose address is 0x80040600 is executed?
Due to the 3-stage ARM pipeline, when an instruction at address A is fetched, the PC is incremented to A+4, then the execution of the instruction begins. Before the execution of the instruction completes, the next instruction is fetched from A+4 and the PC is incremented by 4 to A+8. Hence, when an instruction at address 0x80040600 executes, the PC will contain 0x80040600 +8 = 0x80040608.

10. Explain how the ARM beq instruction behaves differently on the ARM processor compared to how the MIPS beq instruction behaves on the MIPS processor.
The ARM beq instruction employs a 24-bit two's complement displacement that is shifted left 2 bits, sign-extended to 32 bits and added to the updated PC to obtain the branch target address. The branch is taken if the condition codes = 0000.
The MIPS beq instruction employs a 16-bit two's complement displacement that is shifted left 2 bits, sign-extended to 32 bits and added to the updated PC to obtain the branch target address. The branch is taken if the zero flag = 1.