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

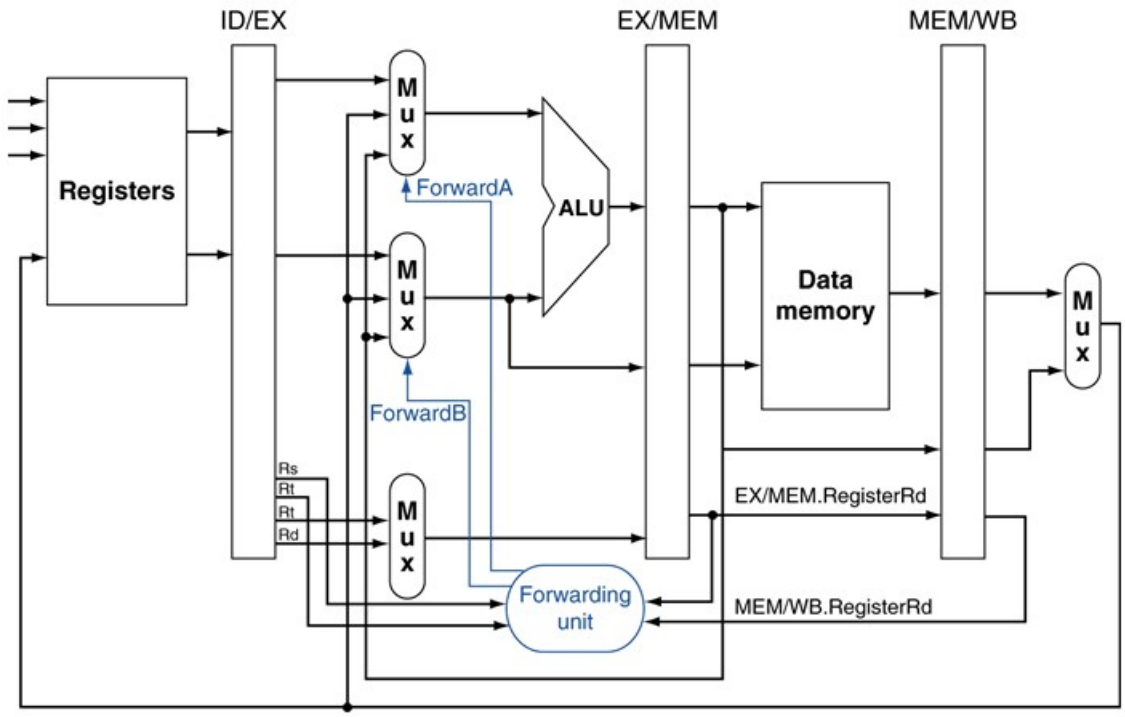
| Page | Frame | Valid Bit |
|---|---|---|
| 0 | – | 0 |
| 1 | 3 | 1 |
| 2 | 0 | 1 |
| 3 | – | 0 |
| 4 | – | 0 |
| 5 | 1 | 1 |
| 6 | 2 | 1 |
| 7 | – | 0 |

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) |

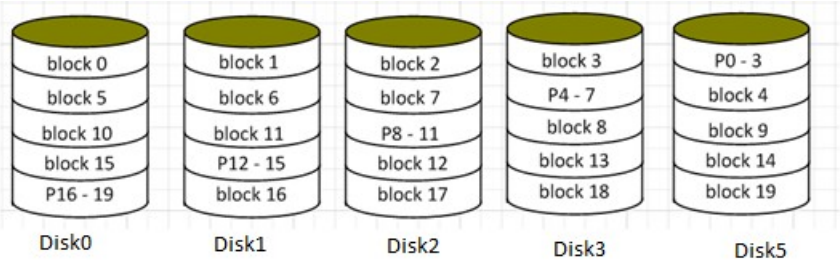



Which one of the following instructions could not be allowed to occupy the execute stage during this clock cycle?

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.