Consider the following code sequence used often in block copies. This produces a special form of the load hazard and normally introduces one stall cycle.

lw $7, 0($12)

sw $7, 0($13)

Show how you can use forwarding to avoid the occurrence of stalls in this situation. Provide your solution by modifying the view of the datapath provided below and precisely state the conditions you would check to implement such forwarding.

Add a multiplexor to the data input to memory. The inputs are as shown at left. Now the output of memory from the previous instruction can be fed to the input to memory for the current instruction.
2 Consider the following instruction sequence with respect to the pipelined datapath with hazard detection, data forwarding, and branch prediction with flushing. Assume writes in WB take place in the first half of the cycle and reads take place in the second half of the cycle. Registers $3, $4, and $5 contain 0x11, 0x22 and 0x88 respectively. Branches are assumed not taken with 38% of conditional branches being taken in practice.

```
lw $4, 0($3)
slt $5, $4, $2
bne $5, $0, Cont
li $4, 0x0400
addi $4, $4, 0x0004
sw $9, 0($4)
```

**Cont:**

2 (a) If the first instruction starts at location 0x10000000, what are the values of the following variables in the pipeline during cycle 4 assuming the branch is taken?

- **IF/ID.PC:** 0x1000000c
- **EX/MEM.aluresult:** 0x11
- **EX/MEM.write.value:** 0x22
- **EX/MEM.wreg.addr:** 4

2 (b) For the case of the branch being taken as well as the case of the branch not taken, at what cycle will the last instruction exit the pipeline?

Counting the first cycle as cycle 1.

On cycle 11 when not taken since there is a stall after the load and branches are assumed not taken.

On cycle 11 when taken since the branch has a single delay slot and the first load has a single delay slot.
3 Assume branch instructions occur 22% of the time and are predicted as not taken, while in practice they are taken 42% of the time with a penalty of 2 cycles. With forwarding, the load delay slot is one cycle and can be filled 55% of the time with useful instructions. 21% of the instructions are loads and 30% of these introduce load delay hazards.

3 (a) What is the increase in CPI due to load delay slots and branch hazards?

\[
\text{load slots + branch delay slots} = 0.21 \times 0.3 \times 0.45 + 0.22 \times 0.42 \times 2 = 0.213
\]

3 (b) To improve performance, we would like to pipeline the memory accesses. This will reduce the clock cycle time by 20%, but increase the load delay slots to 2 cycles while not affecting branch delays. Is this worthwhile? Provide a quantitative justification for your answer.

\[
\text{CPI}_2 = (0.21 \times 0.3 \times 0.45 \times 2) + (0.22 \times 0.42 \times 2) = 0.2415
\]

\[
\text{CPU Time}_1 = I \times \text{CPI}_1 \times \text{clock\_cycle\_time}
\]

\[
\text{CPU Time}_2 = I \times \text{CPI}_2 \times (\text{clock\_cycle\_time} \times 0.8)
\]

The answer reduces to whether

\[
(CPI + 0.213) > (CPI + 0.2415) \times 0.8
\]
Consider the following instruction sequence with respect to the pipelined datapath with load hazard detection, data forwarding, and branch prediction (not taken) with flushing. Assume writes in WB take place in the first half of the cycle and reads take place in the second half of the cycle.

(a) If the first instruction in the above sequence starts at location 0x400, what are the values of the following variables in the pipeline during cycle 8. The first cycle is cycle 1.

<table>
<thead>
<tr>
<th>Both Forwarding Unit Outputs</th>
<th>00, 00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output of the ALU</td>
<td>10024</td>
</tr>
<tr>
<td>IF/ID.PC</td>
<td>41c</td>
</tr>
<tr>
<td>EX.MEM. RegWrite</td>
<td>1</td>
</tr>
<tr>
<td>ID/EX.Register.Rs</td>
<td>10028</td>
</tr>
</tbody>
</table>

The instruction in IF is add $7, $9, $7

(b) From the code, you can determine how many times this loop is executed. Furthermore, remember branches are predicted as not taken (with flushing), and full forwarding and load hazard detection. What is the CPI value that you would obtain if you computed this value only over this block of code? Ignore the pipeline setup time.

72/51 (last time through the loop the branch is not taken)
Suppose a page fault occurred on the data reference for instruction 6. State i) the instructions in the pipeline at the time, ii) those which are allowed to complete execution, and iii) at which instruction does execution resume after the page fault has been serviced, i.e., which is the first instruction to be fetched.

IF - instr 8
ID - instr 7
EX -- bubble
MEM - instr 6
WB ==- instr 5

The above shows the state when the fault occurs. Instruction 5 is allowed to complete and execution will resume with the fetching of instruction 6.
Assume branch instructions occur 15% of the time and are predicted as not taken, while in practice they are taken 40% of the time with a penalty of 3 cycles. With forwarding, the load delay slot is one cycle and can be filled 60% of the time with useful instructions. 20% of the instructions are loads and 30% of these introduce load delay hazards.

(a) What is the increase in CPI due to load delay slots and branch hazards?

\[
\text{CPI} = 1 + \text{increase due to branches} + \text{increase due to loads} \\
= 1 + 0.15 \times 0.4 \times 3 + 0.2 \times 0.3 \times 0.4 \times 1 \\
= 1.204
\]

(b) To improve performance, we would like to pipeline the ALU. This will reduce the clock cycle time by 15%, but increase the load delay slots to 2 cycles, and the branch delays to 4 cycles. Provide a quantitative basis for determining if this is a good tradeoff.

\[
\text{CPI}_{\text{new}} = 1 + 0.15 \times 0.4 \times 4 + 0.2 \times 0.3 \times 0.4 \times 2 = 1.288 \\
\text{Ex}_{\text{new}} = I \times \text{CPI}_{\text{new}} \times 0.85 \times \text{clock\_cycle} = I \times 1.0948 \times \text{clock\_cycle} \\
\text{Ex}_{\text{old}} = I \times \text{CPI}_{\text{old}} \times \text{clock\_cycle} = I \times 1.204 \times \text{clock\_cycle} \\
\]

This is a good trade-off.
Consider a new high speed SPIM processor design where we have an 8 stage pipeline. Relative to the design in the text, the memory stage has been partitioned into 2 stages and the ALU stage has been partitioned into three stages (hence the new 8 stage pipeline). As a result, the clock cycle time has been reduced to 6 ns.

(a) What is the latency experienced by an instruction and the theoretical maximum throughput of this processor in instructions/second? Ignore hazards.

Latency = 48 ns
Maximum throughput is 167 MFLOPS

(b) Now we are examining the next generation design, where we will have two such pipelines operating concurrently, i.e., a superscalar design. An instruction fetched from memory may be executed in either pipeline. Memory is fast enough to supply two instructions in each fetch cycle. Repeat (a) for this pipeline.

Latency is unchanged - each instruction must still go through the entire pipe = 48 ns

The maximum throughput has been doubled since two instructions can complete in each clock cycle = 334 MFLOPS
Consider the state of following program executing within the SPIM pipeline during cycle 6. Assume full hardware support for forwarding, branches predicted as not taken, and flushing when branches are taken. The first instruction is at address 0x400. Assume that the mov instruction is a native instruction.

```
move $6, $0
addi $4, $0, 32
loop:
  lw $10, 1000($4)
  blt $10, $6, next  -- check if $10 < $6
  move $10 $6       -- move $6 to $10
next:
  addi $4, $4, -4
  bne $4, $0, loop
```

7 (a) What instructions are in each stage of the pipeline.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>mov $10, $6</td>
</tr>
<tr>
<td>ID</td>
<td>blt $10, $6, next</td>
</tr>
<tr>
<td>EX</td>
<td>bubble</td>
</tr>
<tr>
<td>MEM</td>
<td>lw $10, 0($4)</td>
</tr>
<tr>
<td>WB</td>
<td>addi $4, $0, 32</td>
</tr>
</tbody>
</table>

7 (b) What are the values of the following fields?

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM/WB.ALUData</td>
<td>0x20</td>
</tr>
<tr>
<td>EX/MEM.MemToReg</td>
<td>1</td>
</tr>
<tr>
<td>IF/ID.PC</td>
<td>0x410</td>
</tr>
<tr>
<td>ID/EX.ALUOp</td>
<td>00</td>
</tr>
</tbody>
</table>

7 (c) Identify all the data and control hazards assuming no forwarding support.

Data Hazards: addi --> lw, lw --> blt, addi --> bne
Control Hazards: blt, bne
8 (a) Consider the state of following procedure executing within the SPIM pipeline during cycle 6. **Assume that the first instruction is fetched in cycle 0!** Assume full hardware support for forwarding, branches predicted as not taken, and flushing when branches are taken. The penalties for hazards are as dictated by the datapath in the figure. Also assume concurrent read/write to the same register in the same cycle. The first instruction is at address 0x00400000. Assume that all instructions are native instructions.

```
lw $t0, 0($a0)
add $a0, $a0,
addi $sp, $sp, -
sw $fp, 4($s1)
addi $fp, $sp,
sw $ra, 0($fp)
sw $t0, -4($fp)
add $a1, $t0,
```

8 (b) (5 pts) What instructions are in each stage of the pipeline.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>sw $t0, -4($fp)</td>
</tr>
<tr>
<td>ID</td>
<td>sw$ra, 0($fp)</td>
</tr>
<tr>
<td>EX</td>
<td>addi $fp, $sp, 16</td>
</tr>
<tr>
<td>MEM</td>
<td>sw $fp, 4($sp)</td>
</tr>
<tr>
<td>WB</td>
<td>addi $sp, $sp, -16</td>
</tr>
</tbody>
</table>

8 (c) (10 pts) What are the values of the following fields? **ALL** multiplexor inputs are numbered from the top to bottom starting at 0. MuxA is the first one from the top in EX.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM/WB.MemToReg</td>
<td>1 (note the convention for labeling)</td>
</tr>
<tr>
<td>ForwardMuxA, ForwardMuxB</td>
<td>01, 00</td>
</tr>
<tr>
<td>IF/ID.PC</td>
<td>0x00400018</td>
</tr>
<tr>
<td>ID/EX.RegWrite</td>
<td>1</td>
</tr>
<tr>
<td>EX/MEM.RegWrite</td>
<td>0</td>
</tr>
<tr>
<td>EX/MEM.WriteRegisterAddr</td>
<td>XX</td>
</tr>
</tbody>
</table>

8 (d) (10 pts) Extensive analysis of the instruction stream of code targeted for the pipelined MIPS with forwarding has produced the following statistics. To support larger caches we have accepted a two cycle access time and therefore the IF and MEM stages have been partitioned into two stages. The alternative is to keep the single cycle hazards but with a clock that is 10% slower. The overall pipeline is now 7 stages. Penalties for load hazards are 2 cycles and those for control hazards are also 2 cycles. We note that 20% of load instructions produce a hazard and 30% of these load delay slots can be filled via instruction scheduling.
Similarly, we can use scheduling can fill 70% of the branch delay slots. Compute the new CPI. If the CPI without additional pipe stages was 1.5, is this design a good idea? Justify your answer.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads</td>
<td>22%</td>
</tr>
<tr>
<td>Stores</td>
<td>13%</td>
</tr>
<tr>
<td>ALU Operations</td>
<td>42%</td>
</tr>
<tr>
<td>Branches</td>
<td>18%</td>
</tr>
</tbody>
</table>

New CPI = 1.5 + 0.22*0.2*0.7*2 + 0.18*0.3*2 = 1.5 + 0.1696 = 1.7
Ex_old = I * 1.5 * 1.1 * clock_cycle_time
Ex_new = I * 1.7 * clock_cycle_time.

Not a good idea.
Consider the trace generated by the pipelined SPIM model that you are working with. The program being executed is the following.

\[
\begin{align*}
\text{add} & \; $1, \; $2, \; $3 \\
\text{add} & \; $1, \; $1, \; $3 \\
\text{sw} & \; $1, \; 0($0) \\
\text{lw} & \; $2, \; 1($0) \\
\text{add} & \; $1, \; $1, \; $1 \\
\end{align*}
\]

9 (a) (3 pts) What are the initial contents of the following?

<table>
<thead>
<tr>
<th>Initial Value in $2</th>
<th>0x44</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value in $3</td>
<td>0x66</td>
</tr>
<tr>
<td>Initial Value in memory word 1 (memory starts from word 0)</td>
<td>0x33</td>
</tr>
</tbody>
</table>

9 (b) (7 pts) From looking at the trace, can you tell if the pipelined model implements forwarding and concurrent read/write? Mark the trace as necessary and clearly justify your answer.

It does not implement forwarding. Look at the first two instructions and successive values of the \( rs \) and \( rt \). Then look at the successive outputs of the ALU as \( ALURes \). Clearly the result of the first addition has not been forwarded to the second instruction.

It does implement concurrent read/write. Look at the time when the last instruction reaches decode. At the same time the second instruction is in in WB. The value being written by the second instruction into $1 is 88 (\( ALU_{WB} \)). The values being read as \( rs \) and \( rt \) by the last instruction on the next cycle show up as 0x88. If concurrent read/write did not happen then the value being read would have been that produced by the first instruction, i.e., 0xAA.
begin
--
-- identify all signals that will show up on the trace and
-- connect them to internal signals on the datapath
--
-- what signals do we want to see from ID?
--
Instr <= id_instruction;
PC <= id_PC4;
--
-- signals traced from EX?
--
Rs <= ex_register_rs;
Rt <= ex_register_rt;
ALUSrc <= ex_ALUSrc;
RegDst <= ex_RegDst;
ALUOp <= ex_ALUOp;
--
-- signals traced from MEM?
--
ALURes <= mem_alu_result;
MemInData <= mem_write_data;
MRead <= mem_MemRead;
MWrite <= mem_MemWrite;
--
-- signals traced from WB?
--
MemOut <= wb_memory_data;
ALU_WB <= wb_alu_result;
WReg <= wb_wreg_addr;
RegWr <= wb_RegWrite;
MemToReg <= wb_MemToReg;

Relevant Text Pieces from spim_pipe.vhd.
Extensive analysis of the instruction stream has produced the following statistics. Data hazards incur a 2 cycle penalty while control hazards incur a 3 cycle penalty. You can leave your answers in expression form.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads</td>
<td>22%</td>
</tr>
<tr>
<td>Stores</td>
<td>13%</td>
</tr>
<tr>
<td>ALU Operations</td>
<td>42%</td>
</tr>
<tr>
<td>Branches</td>
<td>18%</td>
</tr>
</tbody>
</table>

(a) Assuming an ALU or Load/Store instruction has probability of 0.2 of experiencing a data hazard, what is the maximum increase in CPI due to data hazards and control hazards? There is no instruction scheduling and no forwarding.

\[
\text{Max Increase in CPI due to Data Hazards} = (0.42+0.22+0.13) \times 2 \times 0.2 \\
\text{Max increase due to control hazards} = (0.18 \times 3).
\]

(b) Now assume that you have full hardware support for forwarding, and branches predicted as not taken supported by instruction flushing when branches are taken. Branches are assumed to be not taken and instructions are fetched accordingly. Branches are actually taken 44% of the time. 34% of the load instructions produce a single load delay slot which can be filled 60% of the time. What is the improvement in pipeline CPI from (a)?

The new CPI increase is

- Due to Hazards = (0.22 * 0.34 * 0.4) * 1
- Due to control hazards = (0.18 * 0.44 * 3)
10 (c) Rather than using forwarding and hardware support for flushing on branches, suppose that instruction scheduling could successfully fill 70% of the delay slots produced by data and control hazards. Furthermore, the clock cycle time could be reduced by 10% since the multiplexors used for forwarding were on the critical path. Is this option a better one than the use of forwarding? If expressions are not evaluated, show precisely how you can answer this question.

New CPI increase due to data hazards is \((0.42 + 0.22 + 0.13) \times 2 \times 0.2 \times 0.3\)
New CPI increase due to control hazards is \(0.18 \times 3 \times 0.3\)

Execution time is \(I \times CPI_{\text{new}} \times (0.9 \times \text{cycle}_\text{time})\)
Old Execution time is \(I \times CPI_{\text{old}} \times \text{cycle}_\text{time}\).
Compare the two.
Consider the state of following procedure executing within the SPIM pipeline during cycle 8. Assume that the first instruction is fetched in cycle 0! Assume full hardware support for forwarding, branches predicted as not taken, and flushing when branches are taken. The penalties for hazards are as dictated by the datapath in the figure. The first instruction is at address 0x00400000. Assume that all instructions are native instructions.

11 (a) What instructions are in each stage of the pipeline.

func: addi $2, $4, $0
      addi $3, $5, $0
      addi $3, $3, $2
      addi $9, $0, $0

loop: lw $22, 0($2)  -- WB
      hazard cycle -- MEM
      addi $9, $9, $22  -- EX
      addi $2, $2, 4   -- ID
      bne $2, $3, loop -- IF
      hazard cycle
      move $2, $9
      jr $31

11 (b) What are the values of the following fields? Mux inputs are numbered from the top to bottom starting at 0. MuxA is the first one from the top in EX.

MEM/WB.MemToReg 1 (according to text, 0 according to my convention stated above)
ForwardMuxA, ForwardMuxB 00 01
IF/ID.PC 0040001c
ID/EX.RegWrite 1
MEM/MEM.RegWrite 0
MEM/MEM.WriteRegister 0x22

While the nop has de-asserted control signals RegDst is 0 and the rest of the pipeline register has been updated and contains the contents of the add instruction.
11 (c) **If there was no forwarding**, what is the earliest cycle in which the following instructions could execute for the first time in the stage shown. Remember we start counting cycles at cycle 0!

```
add $9, $9, $22 : In EX 11
bne $2, $3, loop: In MEM 16
```
Consider the following code sequence which computes a running sum of an array of integers stored starting at location Start: through memory address End:

```
add $7, $0, $0
la $4, End
la $3, Start
loop:
    lw $6, 0($3)
    lw $5, 4($3)
    add $7, $6, $7
    add $7, $5, $7
    addi $3, $3, 8
    bne $3, $4, loop
``` 

(a) Rewrite the code sequence assuming that the pipelined SPIM provides no support for forwarding or branch hazards and the compiler must ensure correctness by the proper insertion of nops. Assume register file write operations take place in the first half of the cycle and read operations in the second half of the cycle.

```
add $7, $0, $0
la $4, End
la $3, Start
nop
nop
loop:
    lw $6, 0($3)
    lw $5, 4($3)
    nop
    add $7, $6, $7
    nop
    nop
    add $7, $5, $7
    addi $3, $3, 8
    nop
    nop
    bne $3, $4, loop
    nop
```
12 (b) Now schedule the code in 2(a) to remove as many *nops* as possible.

```
la $3, Start
la $4, End
add $7, $0, $0

loop:
  lw $6, 0($3)
  lw $5, 4($3)
  addi $3, $3, 8
  add $7, $6, $7
  nop
  bne $3, $4, loop
  add $7, $5, $7
```
Consider a processor system with 256 kbytes of memory, 64 Kbyte pages, and a 1 Mbyte virtual address space. The processor has an 4 Kbyte direct mapped cache with 256 bytes/line and a 4 entry fully associative TLB. There is also a 16 Kbyte set associative L2 cache with 1024 byte lines, and associativity of 2. The LRU bit is set if the lower line in the set is the least recently used. The state of the memory hierarchy is as shown. Answer the following questions with respect to this memory system.

### TLB

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>c</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>d</td>
<td>10</td>
</tr>
</tbody>
</table>

### Page Table

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Physical Page Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

### TLB LRU Stack

- c
- 0
- d
- 1

### L1 Cache

<table>
<thead>
<tr>
<th>line 0</th>
<th>L2 Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 0</td>
</tr>
<tr>
<td>22</td>
<td>13</td>
</tr>
<tr>
<td>29</td>
<td>09</td>
</tr>
<tr>
<td>0a</td>
<td>0c</td>
</tr>
<tr>
<td>16</td>
<td>0a</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>05</td>
</tr>
<tr>
<td>3c</td>
<td>00</td>
</tr>
<tr>
<td>1a</td>
<td>14</td>
</tr>
<tr>
<td>18</td>
<td>07</td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1d</td>
</tr>
<tr>
<td>11</td>
<td>1e</td>
</tr>
<tr>
<td>2a</td>
<td>08</td>
</tr>
<tr>
<td>10</td>
<td>06</td>
</tr>
<tr>
<td>33</td>
<td>08</td>
</tr>
</tbody>
</table>

### L2 Cache

<table>
<thead>
<tr>
<th>Set 0</th>
<th>L2 LRU Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### L2 LRU Bit

- Main Memory
  - l: 1
  - c: c
  - d: d
- Mem LRU Stack
  - c
  - 0
  - d
  - 1
13 (a) Show the breakdown of the (i) virtual address, (ii) physical address, (iii) physical address as interpreted by the L1 cache, and (iv) the physical address as interpreted by the L2 cache.

<table>
<thead>
<tr>
<th>VA</th>
<th>19</th>
<th>4</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>17</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>PA</td>
<td>17</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>L2</td>
<td>17</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

13 (b) Treat the L2 cache as a main memory for the L1 cache, and consider the wide memory organization for L2 where the width of the bus is now 8 banks. L2 access time is 4 cycles for a bank. Compute the miss penalty from L2 to L1.

One cycle for address transfer, four cycles for memory access and one cycle for transfer of eight words back. The cache line is 64 words. Therefore we have

Penalty = (1 + 4 + 1) * 8 = 48 cycles
13 (c) What is the state of the system after the following sequence of virtual addresses generated by the processors: 0xd9144, 0x09088. You only need show changes on the figure below.

### TLB

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TLB LRU Stack

- 0
- d
- c
- 1

### Page Table

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Physical Page Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### L1 Cache

- 39

### L2 Cache

- 1c

### LRU Bit

- 1

### Main Memory

<table>
<thead>
<tr>
<th>Mem LRU Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
13 (d) Compute the miss penalty in cycles for L2 assuming the main memory is organized with a single word wide bus and interlaved across 32 banks and it takes 20 cycles to access a word.

There are 256 words/cache line. One cycle for address, 20 cycles for memory access time and 32 cycles to transfer the 32 words to the L2. This is repeated 8 times.

Penalty = (1 + 20 + 32) * 8 = 424 cycles

13 (e) Now suppose we have a page fault. The disk has an average seek time of 9 ms and rotates at 7200 rpm. The transfer rate out of the disk is 8 Mbytes/sec and the disk controller overhead is 1 ms. How long does it take to service a page fault?

Servicing a page fault requires transfer of a 64 Kbyte page to memory.

Access time = 1 * 10^-3 + 9 * 10^-3 + (0.5/(7200/60)) + (64/8192)

The expression represents

controller overhead + seek time + rotational delay (average 1/2 rotation) + transfer time.
Suppose we have a 32 x 32 matrix stored in row major form in memory. Consider the initial set of accesses to this matrix which are all of the elements along the main diagonal. How many compulsory misses are there in the L1 cache? How many compulsory misses are there in the L2 cache? Assume that the caches are initially empty. Show your work.

The L1 cache has 256 byte lines which will contain two rows (32 elements in each row at 4 bytes per element storage). Thus a sequential access of elements along the diagonal will produce a miss every two rows = 16 compulsory misses.

The L2 cache has 1024 byte lines each of which can contain 8 lines of the matrix. Thus we will have 4 compulsory misses.

If the local miss rates for the L1 and L2 caches are 1.8% and 36% respectively, and 34% of the instructions are load/store instructions, what is the increase in CPI due to cache misses.

The number of memory references/instruction is 1.34. Each of these references will potentially add cycles as follows.

\[ 1.34 \times (0.018 \times \text{L1 miss penalty} + (0.018 \times 0.036) \times \text{L2 miss penalty}) \]
13 (h) If virtual page 1 is replaced in memory, which lines in the L2 and L1 cache must be flushed to avoid false hits?

Virtual page 1 is in main memory page 00. Therefore all lines from this page will have the upper two bits of the physical address set to 00. The remaining bits can have any value. For L1 this leaves 3 bits in the tag. Thus any line whose tag matches 00XXXX in the L1 cache is from main memory page 00 and therefore from virtual page 1. Similar arguments show that the lines from 00 in the L2 cache correspond to those lines with tags 00XXX.
A computer system has 64Kbyte of main memory, a 256Kbyte virtual address space, a 4Kbyte set-associative cache, and a fully associative 4 entry TLB. The page size is 16 Kbytes, the cache line size is 256 bytes, and there are 4 lines/set. The following shows the state of the complete memory hierarchy. All replacement policies are LRU. Answer the following questions with respect to this hierarchy.

### TLB

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6</td>
<td>11</td>
</tr>
</tbody>
</table>

### Page Table

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Physical Page Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Cache LRU Stack

<table>
<thead>
<tr>
<th>Cache LRU Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>33</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>0D</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>39</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>2E</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>2D</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>23</td>
</tr>
</tbody>
</table>

### Main Memory LRU Stack

<table>
<thead>
<tr>
<th>Main Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

### Mem LRU Stack

<table>
<thead>
<tr>
<th>Mem LRU Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>A</td>
</tr>
</tbody>
</table>
14 (a) Show the breakdown of the fields in the physical address, the virtual address, and the physical address as interpreted by the cache. Draw a figure showing the address fields with their size labeled in bits.

17 15 0
VA  4  2
  14 14

15 15 0
PA  6  2
  2  8

14 (b) What is the relationship if any, between the entries in the TLB LRU stack and the entries in the main memory LRU stack with respect to their values and order of appearance?

The TLB entries should be a subset of the entries in the main memory LRU stack, and in the same relative order.
14 (c) Show the state of the complete memory hierarchy after the following sequence of memory references - 1A672, 28431, 3A623. You only need show changes.

### TLB

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>E</td>
<td>10</td>
</tr>
</tbody>
</table>

### TLB LRU Stack

| E | A | 6 | 0 |

### Page Table

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Physical Page Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>00</td>
</tr>
</tbody>
</table>

### Cache LRU Stack

| 11 | 13 | 28 | 26 |

### Main Memory LRU Stack

| E | A | 6 | 0 | 29 |

### Cache

| 29 | 02 | 29 | 39 | 19 |
15 Consider a direct-mapped cache with 64 byte lines, and a total of 128 lines. The main memory organization is interleaved with a one word wide bus and 4 memory modules. The instruction miss rate is 2% and the data miss rate is 5%. 65% of all memory accesses are for instructions. Memory cycle time is 10 cycles.

15 (a) If the base CPI is 2.2 with a perfect memory system, what is the increase in CPI due to cache misses?

There are 4 words/per access to the interleaved memory modules.

miss penalty = (1 + 10 + 4) * 4 = 60 cycles

The number of memory references/instruction = 1/0.65 = 1.53. Therefore there are 0.53 data references/instruction. Another way of looking at this is that for every 100 references, 65 are to instructions. Therefore memory references/instruction is 100/65.

increase in CPI due to instruction misses is 0.02 * 1 * 60 = 1.20

increase in CPI due to data miss is 0.05 * 0.53 * 60 = 1.59

Total increase is 1.20 + 1.59 = 2.79

15 (b) Is it better to double the clock rate or to redesign the datapath such that the new CPI without cache effects is reduced by half (i.e., to 1.1)? Provide a quantitative justification for your answer.

Reducing base CPI to 1.1 provides an overall CPI of 1.1 + 2.79 = 3.89

Doubling the clock rate results in a CPI of (remember memory access times are unchanged, therefore miss penalty expressed in cycles are doubled)

2.2 + (0.02 * 120) + (0.05 * 0.53 * 120) = 2.2 + 2.40 + 3.18 = 7.78

To compare the two cases we use the expressions for overall execution time. We have

(I*3.89 * clock_cycle_time) / (I*7.78* (clock_cycle_time/2))

There is really no benefit either way.
Consider a 50 MHz processor. Memory accesses over the bus require 1 cycle to accept the address, 3 cycles for memory latency and then 8 cycles to transmit up to 8 words. The latency is reduced to 2 cycles for writes. The processor cache is a write-back cache with 8 word blocks.

(a) If the cache misses are made up of 60% write misses and 40% read misses, what is the maximum bus bandwidth that can be supported between memory and the cache.

To read 8 words requires $1 + 3 + 8 = 12$ cycles, for an average rate of 2.67 bytes/cycle

To write 8 words requires $1 + 2 + 8 = 11$ cycles for an average of 2.9 bytes/cycle

Given the mix, the bus bandwidth can be calculated as

$0.6 \times 2.9 + 0.4 \times 2.67 = 2.8$ bytes/cycle

At 50 Mhz this is $50 \times 10^6 \times 2.8 = 140$ Mbytes/sec

(b) The cache miss rate is 0.05 misses/instruction, and 40% of the misses require the replaced line to be written back to memory. What is the number of cycles per instruction spent handling a miss?

$0.05 \times (12 + 0.4 \times 11) = 0.822$
Consider the pipelined SPIM data path operating with a 20 ns cycle time. The memory hierarchy has the following parameters.

- 32 Kbyte direct mapped L1 unified instruction and data cache with 128 byte lines. The only status bit maintained in the L1 cache directory is the Valid bit. A miss in this cache costs 16 cycles to load from the L2 cache.
- 1 Mbyte, set associative, writeback L2 cache with 128 byte lines and 4 lines/set. The directory maintains a Valid and Dirty bit for each line.
- 256 Mbyte main memory organized as 16 banks, where each bank can deliver 1 word in 20 ns and it is interconnected with the L2 cache by a one word wide bus. It takes one cycle (20 ns) to transmit one word of information across this bus.
- 4 Gbyte virtual address space with 32 K byte pages

Answer the following with respect to this design.

17 (a) Show the breakdown of the virtual and physical addresses as interpreted by the virtual memory system.

17 (b) Show the breakdown of the physical addresses as interpreted by the L1 and L2 caches.
17 (c) What is the size (in bits) of the L1 and L2 cache directories? Show your work.

L1 Directory: [13 (tag) + 1 (valid)] X 256 (#lines)
L2 Directory: [10 (tag) + 1 (valid) + 1 (dirty)] X 8192 (#lines)

17 (d) How many lines are there in main memory that can occupy lines in set 31 in the L2 cache?

This is given by the number of bits in the tag = 1024 lines.

17 (e) What is the miss penalty from the L2 cache in nanoseconds?

\[
\text{mis penalty} = 20 + 20 + 16 \times 20 = 720 \text{ ns.}
\]

17 (f) In the worst case, how many cycles does the pipeline stall on an instruction miss.

The worst case occurs when the instruction is neither in the L1 or L2 cache. Penalty is given by
access time to L1 + L1 Penalty + access time to L2 + L2 penalty
\[
= 20 + 16 \times 20 \text{ ns} + 20 + 720 \text{ ns} = 1080 \text{ ns.}
\]
This is equivalent to 54 cycles. The extra cycle at each level is the time it takes for a hit or to learn that it is a miss.
17 (g) Suppose the following is the contents of a fully associative TLB. What physical address will the reference 0x69540c88 generate?

TLB

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
<th>0x91a0c88</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1B234</td>
<td>0AAA</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0D2A8</td>
<td>1234</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>07640</td>
<td>1456</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1213C</td>
<td>1542</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>12248</td>
<td>087F</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1234F</td>
<td>112C</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>01236</td>
<td>0342</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0B3B3</td>
<td>018C</td>
<td>1</td>
</tr>
</tbody>
</table>

17 (h) If each page table entry contains 32 bits, how large (in bytes) is the page table?

512 Kbytes

17 (i) What is the working set of a program and how does it affect the performance of the virtual memory system?
17 (j) Consider only instruction fetches. If the local miss rate in the L1 cache is 2% and the global miss rate is 1.8% what is the increase is CPI due to instruction misses?

$0.02 \times 16 + 0.018 \times 36$

Note that the L2 penalty is 720 ns or 36 cycles (this expression ignores the single cycle hit time in the L2).
You just purchased a new high workstation and are trying to understand how the cache is organized so that you can effectively layout your data structures. The machine has 256 Mbyte of RAM and a 256Kbyte direct mapped L1 cache. The cache will utilize 128 byte lines. Provide the following.

(a) Show the breakdown of the address bits including the bits used to address bytes within a word, words within a line, line in the cache and the tag. Be sure and indicate how many bits are required of the physical address. Do not assume 32 bit addresses. Provide the number of lines in the cache.

<table>
<thead>
<tr>
<th>27</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

Number of lines in the cache? ______2K = 2048_____

(b) (3 pts) Provide an example of two main memory addresses that will result in the identical tag values in the cache, but correspond to memory lines that will occupy two distinct lines in the cache. Make your answer clear by showing the address breakdown.

Must have different values in the line field.
0x802b30c

0x8039e00

(c) (2 pts) Provide an example of two main memory addresses corresponding to lines that will occupy the same line in the cache. Make your answer clear by showing the address breakdown.

Must have the same values in the line field.
0xad2b308

0xc8ab344

(d) (5 pts) Provide a sequence of three memory addresses such that if the processor contin-
ually cycles through these addresses the cache will eventually start missing on every access.
Must have the same values in the line field and distinct values in the tag field.
0x85eb37c, 0xb4ab33c, 0x182b320
Consider an I/O subsystem structured around a separate memory and I/O buses as discussed in class.

19 (a) Under what conditions would you recommend using an asynchronous communication protocols over a synchronous bus communication protocol?

19 (b) Consider a device that is to transfer data at a rate of 1 Mbyte/sec in 16 byte chunks. The overhead of polling is 400 cycles and the overhead of interrupts is 600 cycles on a 500 Mhz processor. Calculate the overhead of polling and interrupts in servicing this device. When and why would you choose one or the other?

The polling rate is now \((1 \times 2^{20} / 16)\) since service is needed only every 16 byte chunks. The poll consumes 400 cycles for a total of 400 \(* 2^{16}\) cycles. The total number of available cycles is 500 \(* 10^6\) cycles. This represents roughly 5% of the CPU cycles. The number of cycles for servicing an interrupt driven implementation can be similarly computed (the cost is 600 cycles rather than 400 cycles). However, interrupt driven costs are only incurred when there is data to be transferred. Polling costs are always incurred resulting in a permanent loss of 5% of the CPU cycles. Interrupts do incur a greater dollar cost for the system via supporting hardware. Polling is the low cost, inexpensive solution.
20 (a) Consider a virtual memory system with a 1 Gbyte address space, a 256 Mbyte memory and a 32 Kbyte page size. How many entries are there in the page table?
\[ \frac{2^{30}}{2^{15}} = 32K \]

20 (b) You have 512 Kbyte 4-way set-associative cache with 128 byte lines. How many lines in the cache? How many sets?
\[ \frac{2^{19}}{2^7} = 4K \text{ lines} \]
\[ 4K/4 = 1K \text{ sets} \]

20 (c) Which cache has a lower miss rate and why? Direct mapped or fully associative
Fully associative since lines are not constrained to be placed in a specific cache location providing maximum placement flexibility and as a result minimum miss rate.

20 (d) Of the three types of misses, compulsory, capacity, and conflict, which type will you not find in the fully associative cache and why?
conflict misses since placement flexibility is maximum.
21 Consider a processor with the following memory hierarchy and state: 128 Kbyte virtual address space, 64 Kbyte physical address space, 2 Kbyte direct mapped cache, 256 byte lines and 16 Kbyte pages. The machine uses a two entry fully associative TLB. All replacement policies are LRU.

(a) On a page fault let us suppose that physical page 3 is replaced. What lines in the cache should be invalidated?

11XXX
21 (b) Show the state of the memory hierarchy after both of the following references have completed: 0x1A764 and 0x12B88. You only need show the changes.

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Physical Page Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
</tbody>
</table>

21 (c) If the page size were changed to 64 Kbyte, how would the data structures shown in this problem change?

The page table would have only two entries. With only one physical page in memory there would be no LRU stack. The TLB could remain two entries in principle although that would not be desirable.
21 (d) Is the following behavior possible? To receive credit your answer must be substantiated.

A memory reference exhibits the following behavior at each level of the memory hierarchy.

<table>
<thead>
<tr>
<th>TLB:</th>
<th>Memory:</th>
<th>Cache:</th>
</tr>
</thead>
<tbody>
<tr>
<td>hit</td>
<td>fault</td>
<td>Miss</td>
</tr>
<tr>
<td>miss</td>
<td>hit</td>
<td>miss</td>
</tr>
</tbody>
</table>
21 (e) Can the cache start decoding the line address using bits from the virtual address, i.e., before the virtual to physical address translation has completed? Justify your answer by using the breakdown of the address bits.

Yes. The bits of the physical address that are used to specify the cache line are part of the offset, bits 0-13. Note that these bits do not change during virtual to physical address translation. Thus cache line access and address translation can be overlapped.
Consider a 2-way set-associative cache with 64 byte lines, and a total of 256 lines. The main memory organization is interleaved with one word wide bus and 8 memory modules. The instruction miss rate is 2% and the data miss rate is 5%. Instruction profiling demonstrates that 28% of all instructions reference memory. Memory access time is 10 cycles.

(a) If the base CPI is 1.6 with a perfect memory system, what is the CPI when the effect of cache misses is included?

Line size = 16 words.

Miss penalty = \((1 + 10 + 8) \times 2 = 38\) cycles

Per instruction penalty is (instruction miss penalty + data miss penalty)

\[= 0.02 \times 38 + 0.28 \times 0.05 \times 38\]

(b) Suppose we now add an L2 cache with the same line size (the cache in the preceding subproblem is now an L1 cache). The global instruction miss rate is 1.4% and the global data miss rate is 2.2%. The miss penalty from L2 to L1 is 4 cycles for the first word and 2 cycles/word thereafter. Now compute the increase in CPI.

Total increase in CPI is due to misses serviced by L2 and misses serviced by main memory.

Misses serviced by main memory is given by

\[0.014 \times 38 + 0.28 \times 0.22 \times 38\]

Misses serviced by the L2 cache is given by

\[0.02 \times (4 + 30) + 0.28 \times 0.05 \times (4+30)\]
Consider a system with a 4 Kbyte 4-way set-associative cache with 128 bytes lines. Now consider accesses to a 1024 word array where each element is a 32-bit word. When a program accesses this array with a stride of $D$ it means starting with the first element, the program accesses every $D^{th}$ element. For example, for $D=1$ the program accesses every element, for $D=2$ the program accesses every other element, for $D=3$ every third element and so on. Assuming a cache that is initially empty, and a program makes one pass over this array with a stride of $D$, what is the miss rate generated for each of the following values of $D$?

$D = 4$

With 128 byte lines we have 32 words/line. There will 32 misses as there are 32 lines. With $D=4$ there are 8 references/line out of which 7 are hits. The miss rate is $1/8$.

$D = 8$

Again 32 misses. However there are only a total of four references to each line since the program only accesses every 8th word. Therefore the miss rate is $1/4$.

$D = 16$

Following the same argument we compute the miss rate as $1/2$. 
Consider a processor memory system with 128 Kbytes of memory, 32 Kbyte pages, and a 512 Kbyte virtual address space. The state of the memory is as shown. Replacement policy in main memory is LRU. The contents of the main memory data structure is virtual page number of the page occupying that physical page. Similarly the LRU stacks contain virtual page numbers.

24 (a) Show the breakdown of the virtual address

24 (b) Suppose the processor issues the following two virtual addresses in sequence: 0x67440, 0x000AC. Show the state of the memory hierarchy after these both addresses are serviced. Do not distinguish between read vs. write accesses.

see modified data structures above
25 Now consider the addition (to the previous question) of an 8 Kbyte direct mapped cache with 512 bytes/line. The state of the memory hierarchy is as shown. Repeat the previous questions with respect to this memory system.

**Page Table**

<table>
<thead>
<tr>
<th>Line Address</th>
<th>Tag Value (in hex)</th>
<th>Valid</th>
<th>Dirty</th>
<th>Physical Page Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>0001</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Main Memory**

- D
- 0
- 8
- C

**Mem LRU Stack**

- 0
- C
- 8
- D

25 (a) Show the breakdown of the physical address, i.e., which bits are used for addressing within a cache line, address of a line, etc. Show this breakdown in the box below. Be sure to indicate the total number of bits in the physical address by indicating the value of the ‘?’.

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
```

25 (b) Suppose that we chose to construct a 2-way set associative cache instead. The line size remains the same. What would be the number of sets and the number of lines/set?

2 lines/set and 8 sets. The associativity gives you the number of lines/set.
26 A computer system has 128 Kbyte of main memory and an 8 Kbyte set-associative cache. The cache line size is 512 bytes, and there are 4 lines/set. The following shows the state of the cache. All replacement policies are LRU. Answer the following questions with respect to this organization.

26 (a) Can two lines in two different sets have the same value of the tag? Justify your answer by showing the breakdown of the address bits.

Two different line addresses (bits 9-16) can have different values in the set field, but same values of the upper 6 bits, i.e., tag.
26 (b) Show the state of the memory hierarchy after the following sequence of memory references (left to right): 01560, 1B724, 16E40.

![Memory Hierarchy Diagram]

26 (c) Assuming that the cache directory has a single presence bit and one dirty bit, what is the size in bits of the data portion and control portion of the cache.

Data: 8 Kbytes × 8 = 64 Kbits

Control = (1 + 1 + 6) × 16 = 128 bits
Consider a processor memory system with 128 Kbytes of memory, 32 Kbyte pages, and a 512 Kbyte virtual address space. The processor has an 8 Kbyte direct mapped cache with 512 bytes/line and a 2 entry fully associative translation lookaside buffer (TLB) to translate virtual page numbers to physical page numbers. The state of the memory hierarchy is as shown. Replacement policy in main memory and the TLB is LRU. The contents of the main memory data structure is virtual page number of the page occupying that physical page. Similarly the LRU stacks contain virtual page numbers. Answer the following questions with respect to this memory system.

27 (a) Suppose the processor issues the following two virtual addresses in sequence: 0x67440 and 0x000AC (these addresses are in hexadecimal notation). Show the state of the memory hierarchy after these both addresses are serviced.

\[\begin{array}{|c|c|c|c|}
\hline
\text{Valid} & \text{Dirty} & \text{Virtual Page No.} & \text{Physical Page No.} \\
\hline
1 & 1 & C & 11 \\
1 & 1 & 0 & 01 \\
\hline
\end{array}\]

27 (a) Suppose the processor issues the following two virtual addresses in sequence: 0x67440 and 0x000AC (these addresses are in hexadecimal notation). Show the state of the memory hierarchy after these both addresses are serviced.
27 (b) If virtual page D is replaced, which lines in the cache must be flushed to avoid false hits in the cache on references to the new page?

Page number 00XX.
28 Consider a machine with 64 Mbyte of memory and a 4 Gbyte virtual address space. The L1 cache is a direct mapped, 128 Kbyte unified instruction and data cache with 512 byte lines. The L2 cache is a 1 Mbyte, 4-way set associative cache with 512 byte lines. The virtual page size is 16 Kbytes. Answer the following questions

28 (a) How many lines are there in the L1 cache and the L2 cache respectively?

L1 = 256 lines
L2 = 2048 lines

28 (b) Show the breakdown of the i) virtual address, ii) physical address, iii) physical address as interpreted by the L1 cache, and iv) physical address as interpreted by the L2 cache.

![Virtual Address](image1)
![Physical Address](image2)
![Physical Address L1 Cache](image3)
![Physical Address L2 Cache](image4)

28 (c) If each page table entry is 32 bits, what is the page table size in bytes?

$2^{18}$ entries x 4 bytes/entry = 1 Mbyte

28 (d) If main memory is interleaved with 8 banks, each bank can deliver one 32 bit word, and the bus width to memory is 32 bits, what is the miss penalty for the L2 cache? Assume that the access time for a memory bank is 20 cycles and the bus can transmit a 32 bit datum in 1 cycle.

Address cycle time + access time + transfer time =

$16 \times 1 + 16 \times 20 + 128 \times 1 = 464$ cycles
28 (e) Suppose the following is the contents of a 2-way set associative TLB. The LRU bit identifies the least recently used page in a set. Consider the two address sequence of virtual memory addresses: 48D3C244, 3F546220. If there is a miss assume that the virtual page resides in memory in physical page 200. If there is a second miss assume that the physical page resides in memory at address 400. What physical addresses do these references generate?

**TLB**

<table>
<thead>
<tr>
<th>Valid</th>
<th>Dirty</th>
<th>Virtual Page No.</th>
<th>Physical Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2B234</td>
<td>AAA</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0D2A8</td>
<td>234</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>37640</td>
<td>456</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2213C</td>
<td>542</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>22248</td>
<td>87F</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1234F</td>
<td>12C</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>21236</td>
<td>342</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0B3B3</td>
<td>18C</td>
</tr>
</tbody>
</table>

i) 04B0244  
ii) 0802220
Consider a memory system with 16 Mbyte of main memory, and a 256 Kbyte direct mapped cache with 128 byte lines. The main memory system is organized as 4 interleaved memory banks with a 10 cycle access time, where each bank can concurrently deliver one word. The instruction miss rate is 3% and the data miss rate is 4%. We find that 40% of all instructions generate a data memory reference.

(a) What is the miss penalty in cycles?

Address cycles + access cycles + transfer time

\[8 + 8 \times 10 + 32 = 120 \text{ cycles}\]

(b) What is the increase in CPI due to cache misses?

\[0.03 \times 120 + 0.04 \times 0.4 \times 120 = 5.52 \text{ CPI}\]

(c) If we modify the cache to be a fully associative cache we can reduce the instruction miss rate to 2% and the data miss rate to 3%. However, the clock cycle increases by 10%. Clearly show how you could quantify this effect on program execution time to determine if this is a good trade-off.

\[\text{CPI}_2 = \text{CPI}_{\text{base}} + 0.02 \times 120 + 0.03 \times 0.4 \times 120\]

\[= \text{CPI}_{\text{base}} + 3.84\]

\[\text{Exec}_1 = I \times (\text{CPI}_{\text{base}} + 5.52) \times \text{Clock}_\text{Cycle}\]

\[\text{Exec}_2 = I \times (\text{CPI}_{\text{base}} + 3.84) \times 1.1 \times \text{Clock}_\text{Cycle}\]
29 (d) Normally we calculate the effect of pipeline and memory system stalls on the CPI by calculating the total number of additional cycles generated by undesirable events, and amortizing this across all instructions, i.e., compute the overhead on a per instruction basis. One thing we have ignored is the effect of TLB misses. Provide an expression demonstrating how you would compute the effect of TLB misses. Use the expressions for computing CPI increases due to cache misses as a guide.

Computed no differently from the effects of cache misses.

num of memory references/instr * TLB miss rate * TLB miss penalty

TLB miss rates are generally very low in the vicinity of 0.1% - 0.0001%.
30 Consider a 64x64 matrix of 32 bit integers stored in memory in row major order, i.e., all elements of the first row are stored in successive locations followed by elements of the second row, and so on. This matrix is immediately followed in memory by a 64 element vector. The computer system you are running on provides you with 1 Kbytes of physical memory. You need to choose between two pages sizes of 64 bytes and 512 bytes. Consider only accesses to data elements, i.e., ignore instruction fetches, partial results, etc. Which page size produces a lower number of page faults when computing a matrix vector product? (Compute the number of page faults for each page size!).

Total storage requirements is $64 \times 64 \times 4 + 256 \text{ bytes} = 16 \text{Kbytes} + 256 \text{ bytes}$

Stick to the basic algorithm: inner product between each row and the vector.


Total number of page faults is $8 + 63 \times 4 = 260$.

ii) 512 byte pages: Two successive rows fit in one page and the vector occupies a part of one page. Main memory can only support two pages. One will be the vector that is always resident. The second will be two rows of the matrix. This will generate $2 + 31 = 33$ pages faults.
31. Consider the complete memory hierarchy. You have a segmented, paged memory system. Each segment has a maximum size of 256 Mbytes. The processor has 8 Mbytes of memory and a 256 Mbyte virtual address space with 4 Kbyte pages. The cache is a 32 Kbyte, 4-way set associative cache with 64 byte lines. Address translation is supported by a 4 entry TLB. Page table entries are 32 bits. Segment descriptors are 8 bytes and segment selectors are two bytes each. Each process can have up to 4 K segments.

31 (a) Show the breakdown of the fields of the (segment) address, linear address (virtual address), physical address, and the physical address as interpreted by the cache.

31 (b) Fill in the following:

- Number of cache lines/page: 64
- Number of virtual pages/segment: 64K
- Number of entries in the Page Table: 64K
- Size of the segment descriptor Table: 512 Kbytes or 64 Kbytes (Intel)

31 (c) What is the difference between a linear address and a virtual address?

A linear address may in fact be a physical address. Segments do not have to be paged, i.e., implemented as paged virtual memory.