This exam has 14 pages; the final sheets are provided as reference and scratch paper, but must be handed in with the exam.

Your scantron will be used for three groups of 10 questions, for a total of 30 responses on the scantron.

You have 120 minutes.

No calculators or other electronics are allowed. You may bring one 8.5” x 11” sheet of handwritten notes that must be handed in with the exam.

To make sure you receive credit, please write clearly and show your work.

We will not answer questions regarding course material.

<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum</th>
<th>Your Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Multiple choice</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: Pipelining (40 points)

Above is the pipeline discussed in class, except with forwarding from the WB stage removed. Unconditional jumps are resolved in ID. Conditional branches are predicted not taken and resolved in EX. Register writes takes place in the first half and reads in the second half of the cycle.

**Part (a)** The processor is currently executing the code shown to the right. In the current cycle, the *lw* instruction is in the **WB** stage. Compute the control signals (marked 1-10 on the diagram) for this code sequence and enter your answers on the scantron. For this question, use the following key: a) 0, b) 1, c) 2, d) 3, e) don’t care/can’t know. (10 points)

```
func:  lw   $t0, 0($a0)
      bne  $t0, 0, A  # taken 80%  #11
      move $v0, $t0
      j    end

A:  lw   $t1, 0($t0)  #13
    li    $t0, 0       #14
    bge  $t1, $a1, skip  # taken 20%  #15
    li    $t0, 1
skip: add  $t2, $t0, $t1  #16
      add  $t3, $t2, $a2  #17
      add  $v0, $t2, $t3  #18
      sw   $t2, 0($a0)   #19
      add  $v1, $t3, $a3  #20
end:  jr   $ra
```
Question 1: Pipelining (cont.)

For all of this page use the MIPS assembly code and the datapath on the previous page.

(b) Annotate the MIPS assembly to indicate all of the true data dependences. (5 points)

(c) Indicate (using #11 through #20 on the scantron) whether each instruction labeled with a number stalls relative to the immediately proceeding instruction. For each numbered instruction enter: (a) no new stall, (b) a 1-cycle stall, (c) a 2-cycle stall. (5 points)

(d) Calculate the average number of cycles the function `func` takes to run from the instruction fetch (IF) stage of the first instruction to the writeback (WB) of the last instruction. Explain. Feel free to use the grid at the end of this exam to aid in your computation, but you should put a full explanation here. (10 points)

(e) Rewrite/reschedule the function `func` to make it faster. Your new code should produce identical results as the original code and you shouldn’t change the interface (e.g., the arguments/return value). (10 points)
Question 2: Cache Analysis (25 points)

Consider the following loop nest along with a 128KB direct-mapped, write-back L1 cache with 64B blocks on a machine with 32-bit address spaces (both virtual and physical) and no hardware prefetching.

Assume that \( \text{sum} \), \( i \), \( j \), and \( k \) are allocated to registers for the duration of the code. The \( \text{pow()} \), \( \text{polynomial()} \), and \( \text{reduce()} \) functions do not perform any memory accesses. Assume that the memory instructions are executed in the order specified by the program and that addresses translation maps virtual addresses to the same physical addresses. You should also assume that arrays \( A \) and \( B \) are contiguous in memory.

```c
const int length = 24 * 1024;
int A[length], B[length]; // ints are 4 bytes big
int sum;

for (int i = 0; i < 4; i++) {
    for (int j = 0; j < length; j++) {
        sum += polynomial(A[j], i + 1);
    }
    for (int k = 0; k < length; k++) {
        sum += reduce(B[k], pow(i, 2));
    }
}
```

**Part (a)** Compute the number of misses for \( i = 0 \), that is, for the first iteration of the outermost loop (and explain how you computed it!). (10 Points)
Question 2: Cache Analysis (cont.)

Part (b) Compute the number of misses for $i = 1$, that is, for the second iteration of the outermost loop (also explain how you computed it). (10 Points)

Part (c) Compute the number of misses for $i = 1$ assuming the cache is fully associative (instead of direct mapped). Assume the cache uses an LRU (Least Recently Used) replacement policy. Explain. (5 Points)
Question 3: Cache-aware Programming (25 points)

(a) Rewrite the following to optimize the cache performance of the code on a system with no support for prefetching and a single-level cache. The cache is a 2-way set associative 32KB cache with 32B blocks. Ints and floats are 4 byte data types. Assume that the hills array starts at an address divisible by 4096. Explain the transformations you applied and why. Your score will depend both on how much you optimize and how well you explain. (15 points)

```c
#define NUM_ANT_TYPES 16
#define DIRECTIONS 8
#define NUM_HILLS_RECORD 81000

struct anthill {
    float longitude, latitude;
    int tunnels[DIRECTIONS];
    int ant_count[NUM_ANT_TYPES];
    char loc_name[24];
};

struct anthill hills[NUM_HILLS_RECORD]; // filled in elsewhere
int total_ants[NUM_ANT_TYPES];

for (int i = 0; i < NUM_ANT_TYPES; i++) {
    total_ants[i] = 0;
    for (int j = 0; j < NUM_HILLS_RECORD; j++) {
        total_ants[i] += hills[j].ant_count[i];
    }
}
```
Question 3: Cache-aware Programming (cont.)

(b) Apply tiling to the following code example to optimize cache behavior. Assume the L1 cache is a 2-way set associative 32KB cache with 32B blocks. Double precision floating point numbers are 8 byte data types. Explain your choice of tile size. (10 points)

\[
\text{const int } N = 10000; \\
\text{double A[N][N], B[N][N];}
\]

\[
\text{for (int } i = 0 ; i < N ; i ++) { \\
\quad \text{for (int } j = 0 ; j < N ; j ++) { \\
\quad \quad B[i][j] = A[j][i]; \\
\quad }
\}
\]
Multiple Choice (10 points)

Choose the best answer and answer on the provided scantron.

21) If everything else is equal, a higher CPI means better performance.  
   A) TRUE    B) FALSE

22) Pipelining improves:  
   A) Both latency and throughput  
   B) Latency, but not throughput  
   C) Throughput, but not latency  
   D) Neither latency nor throughput

For a 32KB 2-way set-associative, write-back cache with 32B blocks on a machine with 32-bit address spaces (both virtual and physical): (for questions 23-24)

23) How many bits is the block offset?  
   a) 5  
   b) 6  
   c) 7  
   d) 8  
   e) 9

24) How many bits is the index?  
   a) 5  
   b) 6  
   c) 7  
   d) 8  
   e) 9

25) In real machines, the least recently used policy works because it takes advantage of:  
   A) spatial and temporal locality in programs   B) spatial locality in programs (not temporal)   
   C) temporal locality in programs (not spatial)   D) neither spatial nor temporal locality

26) Which of the following is false, or select (e) if all of the statements are true:  
   a. Accessing multi-level page tables involves using a different portion of the physical page number as the index for each level of the page table.  
   b. A page fault is when the desired page is not currently mapped into memory.  
   c. The page table base pointer can only be written by the operating system.  
   d. Virtual memory systems rely on both spatial and temporal locality for efficiency.  
   e. all of the statements are true.

27) When reading a whole track from a disk, what is likely to be the largest component of the read time:  
   A) rotational latency  
   B) circulation latency  
   C) transfer time  
   D) escape time  
   E) seek time

28) What kind of prefetching can be used to prefetch for the following code:

```c
for (int i = 1; i < (LENGTH - 1); i ++) {
    A[i] = B[i-1] * C[i+1]
}
```

   A) hardware prefetching only  
   B) software prefetching only  
   C) either hardware or software prefetching  
   D) neither hardware nor software prefetching
For the following pieces of code, indicate which forms of locality are definitely exhibited for the accesses to the bolded data structure:

a) temporal locality  b) spatial locality  c) both  d) neither

29) for (int i = 0 ; i < NUM ; i ++) {
    for (int j = 0 ; j < NUM ; j ++) {
        elements[i][j] = starting_elements[i + j];
    }
}

30) for (int z = 0 ; z < 4096; z += 16) {
    rays[z] = base[z] + flare[z];
}