CONFIDENTIAL

Paper 3.32=E4.53=F2.25=ISE3.9c Parallel Architectures

Department of Computing
Imperial College

Examination paper **March 22, 1995**

First examiner: .... **Second examiner:** ....

1 Modern microprocessors stall if the memory system fails to provide an operand within one or two cycles. Consider the following architectural techniques which have been proposed to alleviate the stalls which arise when this is not possible:

a Static pipeline architecture with instruction scheduling by the compiler — i.e. the hardware implements a straightforward load instruction which delivers the operand to the specified register within some maximum number of instructions, and the compiler aims to find useful instructions to fill this delay.

b Prefetching. That is, provide a special instruction which preloads the cache with a value which is expected to be needed soon.

c Vector instructions and interleaved memory, for example as found in vector pipeline machines like DLXV and the Cray 1.

d Multithreading, that is use each processor to run a small set of processes. When one blocks on access to memory, switch to another.

Make *very brief* notes on each of these approaches, paying special attention to the costs of the scheme (both hardware and software), the limits on its effectiveness, and its impact, if any, on cache design.

2a When a write miss occurs, two policies are possible: a cache line could be allocated and updated with the value written (called *allocate-on-write*), or alternatively the cache could be left unchanged and the value could be written direct to main memory *(no-allocate-on-write)*.

Give a small example high-level language program which should run substantially faster using no-allocate-on-write than using allocate-on-write. Explain.

b Some machines (e.g. the Intel i860) have two kinds of load instructions, one which allocates on load, the other which doesn't. Give a small example high-level language program which should run substantially faster using such a no-allocate-on-load instruction than using an allocate-on-load instruction. Explain.

c A computer architect is considering two alternative designs, which turn out to have similar costs:

A A static-pipeline DLX CPU, 4ns clock and a single, 128K byte cache accessible with no pipeline stalls.

B A static-pipeline DLX CPU, 2ns clock, with a 4K byte primary cache accessible with no pipeline stalls, and a 64K byte secondary cache accessible with two pipeline stall cycles.

The two designs have a similar main memory system, with a 50ns access time. The application software displays the following characteristics:

- Assuming no memory delays, the CPI is 1.5.
- 50% of instructions involve memory access.
- 25% of memory accesses result in misses in the 4K byte primary cache which are hits in the secondary cache.
- 2% of memory accesses result in misses in the 64K byte secondary cache.
- 1% of memory accesses result in misses in the 128K byte cache.

Compute the instruction processing rate (MIPS) for the two designs. Show your working and state any assumptions you need to make.

(The three parts carry, respectively, 20%, 20% and 60% of the marks).

Turn over ...

3 Cache consistency (sometimes called cache coherency) concerns the problem of ensuring that the correct value is always used, when caching leads to multiple copies of a value.

Describe as briefly as possible the cache consistency issues in the following situations (some of them do overlap—explain):

a Hardware support for message passing.

b Self-modifying code.

c A write-back FIFO buffer, which can queue five words for writing before a write stall need occur even on a cache miss.

d Virtual memory (suppose that the cache is indexed using a virtual address, and uses virtual addresses as tags).

Where possible, suggest how the consistency problem might be solved.

4 Consider the following code fragment:

```
for i = 1 to 4
  for j = 1 to 4
S₁  A[i,j] := A[i,j] * 2;
  end
  for j = 1 to 4
S₂  B[i,j] := A[i,j+1] + A[i,j-1];
  end
end
```

a Draw the iteration space for this loop, showing all instances of both S_1^{ij} and S_2^{ij}. Indicate the execution order specified by the program.

b Mark your iteration space diagram to show all dependences.

c Discuss whether it is possible to fuse the two inner loops.

d Consider the following loop nest:

```
for i = 1 to N
  for j = 1 to N
S₁  A[i,j] := A[i,j] * 2;
  end
  for j = 1 to 4
S₂  B[i,j] := A[i,j+1] + A[i,j-1];
  end
end
```

Consider a distributed-memory MIMD machine with P processors, connected by a message-passing interconnection network (`N` much larger than P).

Assume that the arrays `A` and `B` are distributed *columnwise* over the processors with blocksize $K = N/P$, so that each row `A[i,0:N+1]` and `B[i,0:N+1]` is distributed across the processors.

Give pseudo-code for a typical processor, showing where message passing is required using the functions `send` and `recv`:

- `send(m,X[L:U])`
  Send a message to PE$_{m)}$ containing the array elements `X[L]`, `X[L+1]`, ... `X[U-1]` and `X[U]`.
- `recv(m,X[L:U])`
  Receive a message from PE$_{m)}$ and store it in the array locations `X[L]`, `X[L+1]`, ... `X[U-1]` and `X[U]`.

(The four parts carry, respectively, 10%, 20%, 30% and 40% of the marks).