

Department of Computer Science
Computer Architecture Qualifying Exam
August 20, 2004

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- show your work whenever appropriate. There can be no partial credit unless we see how answers were derived.
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

1. (40 points) A two-operand computer has a macro-instruction set with instructions of the following forms:

op R1 R2	Register-register (RR) instructions
op R1 index(R2)	Register-memory (RM) instructions
op R const	Register-immediate (RI) instructions

The instructions are `add`, `sub`, `lod`, and `sto`, which all have their expected meanings: `add` adds a value to a register, `sub` subtracts a value from a register, `lod` loads a new value into a register, and `sto` stores a value to memory. There are two registers visible to macro-instructions, named A and B. All the addressing modes that make sense for an instruction are available to it, so an example of a possible code sequence might be

```
lod A, 100 # load a value of 100 into A
add A, 37(B) # add the contents of memory location 37+B into A
sto A, 18(B) # store the contents of A to memory location 18+B
```

These instructions are translated into μ ops, much as in the P6. The μ ops have the following forms:

```
 $\mu$ op R1 R2
ldi R1 const
```

The μ ops are `ad`, `sb`, `ld`, `ldi` and `st`, and also have their expected meanings: `ad` adds two registers together or adds a constant to a register, `sb` subtracts, `ld` loads the contents of a memory location into a register, `ldi` loads an immediate value into a register, and `st` stores the contents of a register to a memory location. You'll notice that `ldi` is the only μ op which can use an immediate value. There are three micro-architectural registers, A, B and T.

- (a) Translate the macro-architecture instruction sequence given as an example above into μ ops.
- (b) Draw a picture of a likely superscalar pipeline for this machine.

2. (25 points) In DEC's original paper introducing the Alpha microprocessor, the following statement appears:

An additional problem with byte stores is that an implementer may easily choose only two of the three design features: fast write-back cache, single-bit error correction code (ECC), or byte stores.

Why is this true?

3. (20 points) A computer system with a 64 bit virtual address space and a 48 bit physical address space and a 4K byte page size uses a hashed page table. Assume the page table contains only eight entries, and is indexed by the last three bits of the page number (this is ridiculously small, of course). The page table contents are:

Index	VPN	PFN	Next
0	17839ac578a98	000003478	3
1	ac893b4f7c169	00000010a	
2	998adf0312364	00001acf8	6
3	889bcfe125678	00000ac12	
4	0003578ff69ac	000000123	2
5	8aacbff897832	000011acd	
6	897fbac10bdc2	0000ff120	7
7	deadbeef12344	00001acdf	

Protection information is not shown. Entries whose “Next” column are empty are at the end of hash collision chains.

For each of the following virtual addresses, describe what happens when attempt is made to translate to a physical address. Be sure to tell each of the table entries that is inspected, both in the case of success and failure.

- (a) 17839ac578a98a3f
- (b) 897fbac10bdc2a37
- (c) ac12f143d1234aac

4. (15 points) In some computer system, an invalidate-based MESI snoopy cache is used to maintain consistency. Four processors make a sequence of memory accesses, as shown in the following figure.

P_1	$W(x)_3$	$W(z)_1$	$R(x)_3$
P_2		$W(y)_2$	$R(z)_1$
P_3	$R(x)_3$		
P_4		$W(y)_1$	

x and y, and z are each in different cache blocks.

If you assume that a processor which holds a block in the Modified state Invalidates that block whenever another processor obtains it, what are the states of the cache blocks in the processors’ caches at the end of that sequence of operations?