V2

October 10, 1996

# cs470 - Computer Architecture 1
**Fall 1996**

## Midterm Exam
open books, open notes

Starts: **6:20 pm** Ends**: 8:00 pm**

Name:_____(please print)

ID:_____

| Problem | Max points | Your mark | Comments |
|---|---|---|---|
| 1 | 20 | | 10+5+5 |
| 2 | 45 | | 10+25+5+5 |
| 3 | 15 | | 15 |
| 4 | 5 | | The first 'do at home' problem |
| 5 | 5 | | The second 'do at home' problem |
| | 90 | | |

**1.** Your first job as a new employee with ACME Computing is to upgrade their printer. Right now it has no memory and their computer has to spend, on the average, half the time of a job session just to feed the printer. You plan to add some memory as to increase printing speed. 4MB of memory will make printing by 90% faster and will increse the cost of the whole system by 1.5%. Your boss wants to know:

a) by how much faster job sessions will be;

b) how the new job throughput will compare with the old one;

c) whether the new system will be more cost effective than the current one.


© Virgil Bistriceanu, 1996

**2.** Consider the following C statement:

```
a = a - b*a;
```

The right hand side can be put in reverse Polish notation as:

```
aba*-
```

You have an 8-bit stack machine with 16 bit addresses; the only addressing mode is absolute. `a` and `b` are of type `very short int` (i.e. 8-bit) and are stored in memory starting with the address `0xDDD` (where DDD stands for the first three digits of your SSN). Their initial values are `0x2b` for `a` and `0x02` for `b`. The code starts in memory at address `0x5ffc`. Below is a list of opcodes (each opcode is one byte wide):

| Instruction | Opcode |
|---|---|
| `add` | `0x20` |
| `sub` | `0x21` |
| `mul` | `0x22` |
| `div` | `0x23` |
| `push` | `0x24` |
| `pop` | `0x25` |

a) Compile the C code for this stack machine; for each instruction in the program show the instruction format on the right hand sid; clearly mark the boundaries of each instruction.

| Instruction | Instruction Format | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |


© Virgil Bistriceanu, 1996

b) Show the sequence of addresses issued by the CPU to execute this code; for each address indicate whether it's a memory read or write and what's on the data bus for that particular memory access.

| Address | R/W | Data | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |


© Virgil Bistriceanu, 1996

c) Assuming that each memory access takes on the average 2.5 clock cycles and that the execution phase of an arithmetic or logic instruction takes 2 clock cycles, find the average CPI of an instruction using the assembly program you wrote.

d) What is the average number of memory accesses per instruction?

**3.** In some simple cases a simple instruction set like MIPS can synthesize powerful instructions found in richer instruction sets like the VAX. The following instruction uses a combination of base+displacement and indirect memory addressing

```
ADDB @(R0), @5(R1)      # @5(R1) stands for M[M[R1]+5]; B in ADDB stands
                        # for Byte
```

Show the sequence of *native* MIPS instructions that would do the same job; use registers `$t0` and `$t1` to represent `R0` and `R1` respectively.


© Virgil Bistriceanu, 1996