

CSE Qualifying Exam, Fall 2006

October 21, 2006

Architecture

1. Consider the following code fragment from an if-then-else statement of the form

```
if (A==0) A = B; else A = A+4;
```

where A is at 0(R3) and B is at 0(R2):

```
LD      R1,0(R3)      ; load A
BNEZ    R1, L1        ; test A
LD      R1, 0(R2)     ; then clause
J       L2            ; skip else
L1:     DADDI R1,R1,#4 ; else clause
L2:     SD      0(R3),R1 ; store A
```

For the following question, assume a standard single-issue MIPS pipeline, branch resolution in the ID stage, delayed branches, and forwarding.

Assume conditional load instructions

```
LWZ Rd,x(Rs1),Rs2 and
LWNZ Rd,x(Rs1),Rs2
```

that do not load unless the value of Rs2 is zero or not zero, respectively. Compile the code using a conditional load, and write it showing any stall cycles that would occur in the pipeline. Compare the clock cycles and register use to that of the original code fragment.

2. Suppose we have an application running on a 32-processor multiprocessor, which has a 400 ns time to handle reference to a remote memory. For this application, assume that all the references except those involving communication hit in the local memory hierarchy. Processors are stalled on a remote request, and the processor clock rate is 1 GHz. If the base CPI (assuming that all references hit in the cache) is 2, how much faster is the multiprocessor if there is no communication versus if 0.2% of the instructions involve a remote communications reference?
3. Recall that memory interleaving allows for a sequence of words in memory to be accessed sequentially, requiring only a single memory access for the sequence (instead a separate memory access for every word). Consider the following description of a computer and its cache performance:

Computer:

Memory accesses per instruction = 1.2

Average cycles per instruction (ignoring cache misses) = 2

Cache:

Block size = 1 word

Miss rate = 3%

Cache miss penalty = 64 cycles (broken down under Memory)

Memory:

Memory bus width = 1 word

Access times:

- 4 clock cycles to send the address
- 56 clock cycles for the access time per word
- 4 clock cycles to send a word of data

If we change the block size to 2 words, the miss rate falls to 2%, and a 4-word block has a miss rate of 1.2%. What is the improvement in performance of interleaving two ways and four ways versus doubling the width of memory and the bus? (Hint: since the clock cycle time and instruction count won't change, calculate performance improvement using CPI.)

Algorithms

- Let A be an array containing n integers. Write an algorithm to find the maximum sum for a contiguous subsequence of elements of A . (If all elements are negative, the maximum subsequence is defined as the empty subsequence with the sum of zero.) For example, for the array

index	1	2	3	4	5	6	7	8	9	10	11	12
value	38	-62	47	-33	28	13	-18	-46	8	21	12	-53

the maximum contiguous subsequence sum is 55 in positions 3 through 6. You will get credit based on the complexity of your algorithm (assuming it is correct) as follows:

complexity	credit
$O(n)$	100%
$O(n \log n)$	80%
$O(n^2)$	40%
$\omega(n^2)$	0%

- Let $G = (V, E)$ be a digraph with nonnegative-integer edge weights given by $c : E \rightarrow \mathbb{N}$. Recall Dijkstra's single-source shortest path algorithm:

DIJKSTRA(v)

Let Q be an empty min priority queue

$d[v] \leftarrow 0$

$\pi[v] \leftarrow \text{null}$

Insert v into Q (the d -value is the key)

For all vertices $w \neq v$, do

$d[w] \leftarrow \infty$

Insert w into Q (the d -value is the key)

End for

While Q is not empty, do

Remove a vertex x with minimum d -value from Q

For each y such that $(x, y) \in E$, do

If y is not in Q , then go on to the next y

Else, if $d[y] > d[x] + c[x, y]$ then

Decrease the key $d[y]$ of y to $d[x] + c[x, y]$ in Q

$\pi[y] \leftarrow x$

End if-else-if

End for

End while

End DIJKSTRA

Modify Dijkstra's algorithm to find, for each vertex w , *all* edges that lie along shortest paths from v to w (there can be more than one path). What additional data structure(s), if any, do you use?

- Let NON-UNIQUE-SAT be the following decision problem:

Instance: A Boolean formula φ in conjunctive normal form (CNF).

Question: Does φ have at least two distinct satisfying assignments?

Show that NON-UNIQUE-SAT is NP-complete by

- (a) (20% credit) showing that NON-UNIQUE-SAT is in NP, and
- (b) (80% credit) describing a polynomial reduction from CNF-SAT to NON-UNIQUE-SAT.

Compilers

1. In a certain simulation language it is convenient to execute a randomly chosen statement from a list of statements.

The following code is an example:

```
choice
  res = f1(x,y,z);
  res = f2(y,x,w);
  if (x < y) then res = f3(x,z)
  else res = f4(y,z);
endchoices;
```

When this code is executed, one of the three statements (two assignment statements and an if-then-else statement) is randomly selected and executed.

Assume that there is a library function `random(n)` that will return a random integer x in the range $[1, n]$ (from 1 to n inclusive). That is, assume there is a 3-address code instruction " $x := \text{random}(n)$ ".

- (a) (30% credit) Give a grammar for handling this type of statement that will handle an arbitrary number of statements as "choices."
 - (b) (10% credit) What attributes are needed to support this?
 - (c) (20% credit) Show the code that would be generated for the example above.
 - (d) (40% credit) Give semantic actions to handle the generation of intermediate code (3-address code, or quadruples if you prefer) for the `choice` statement.
2. **Activation records:** Given the code segment below:

```
main(){
  ...
  x = f(y, z*3);
  ...
}
float f (int px, int py){
float x;
float y;
...
}
```

- (a) (25% credit) Show the contents of the stack just before the call to `f`.
 - (b) (25% credit) What actions must be taken in the prologue? Show the contents of the stack just after the prologue has completed.
 - (c) (25% credit) What would the offset be for each local variable and each parameter?
 - (d) (25% credit) What actions must be taken in the epilogue?
3. Consider the intermediate code below.

```

0    sum = 0
1    i = 0
L0: 2    j = 0
L1: 3    t1 = b
      4    t1 = t1 + i * w
      5    t1 = t1 + j * 8
      6    f = array [ t1 ]
      7    if (f > 0) goto L2
      8    goto L3
L2: 9    sum = sum + f
     10   goto L4
L3: 11   sum = sum - 1
L4: 12   j = j + 1
     13   if (j < 32) goto L1
     14   i = i + 1
L5: 15   if (i < 8) goto L0
L6: 16   return
L7: 17   goto L0

```

Assume that there are no entry points into the code from outside other than at the start.

- (20% credit) Decompose the code into basic blocks B1,B2, . . . , giving a range of line numbers for each.
- (20% credit) Draw the control flow graph, and describe any unreachable code.
- (40% credit) Fill in an 18-row table listing which variables are live at which control points. Treat `array` as a single variable. Assume that `n` and `sum` are the only live variables immediately before line 16 (the only exit point). Your table should look like this:

Before line	Live variables
0	...
1	...
2	...
...	...
17	...

- (20% credit) Describe any simplifying transformations that can be performed on the code (i.e., transformations that preserve the semantics but reduce (i) the complexity of an instruction, (ii) the number of instructions, (iii) the number of branches, or (iv) the number of variables).

Theory

1. For any binary string $x \in \{0, 1\}^*$, we let \bar{x} denote the nonnegative integer that x represents in binary. (Just to avoid any confusion, here is an inductive definition of \bar{x} : $\bar{\epsilon} = 0$; $\overline{x0} = 2\bar{x}$; $\overline{x1} = 2\bar{x} + 1$.) Let $A \subseteq \{0, 1\}^*$ be a language, and let n be a positive integer. Define the language

$$A + n := \{x \in \{0, 1\}^* : (\exists y \in A)[|x| = |y| \text{ and } \bar{x} = \bar{y} + n]\}.$$

- (a) (75% credit) Show that if $A \subseteq \{0, 1\}^*$ is regular, then $A + 1$ is regular. [Hint: Given a k -state DFA or NFA for A , you can construct an NFA for $A + 1$ with $2k$ states.]
 - (b) (25% credit) Assuming the result of part 1a, prove that if A is regular then $A + n$ is regular for all positive integers n .
2. Show that every infinite Turing-recognizable language includes (as a subset) an infinite decidable language. That is, for any infinite Turing-recognizable language L , there is an infinite decidable language A such that $A \subseteq L$. [Hint: It may help to consider an *enumerator* for L , i.e., a Turing machine that runs forever and outputs strings x_1, x_2, x_3, \dots from time to time such that $L = \{x_1, x_2, x_3, \dots\}$. You may assume the fact that every Turing-recognizable language has an enumerator.]
 3. Show that if $P = NP$, then there exists a deterministic polynomial-time algorithm A that takes as input a CNF Boolean formula φ and outputs a satisfying assignment for φ if one exists, and outputs “unsatisfiable” otherwise.