Fall 2015 CSE Qualifying Exam
Core Subjects

September 26, 2015

Architecture

1. Assuming the classical 5-stage pipeline with no forwarding except through the registers and all operations execute in one cycle. Assume you predict branch not taken and then branch is taken.

   Now Assume the following differences
   Given the code below:

```
loop:  LD     R6, 0(R1)
       ADD    R5, R6, R6
       ADD    R10, R6, R5
       ADD    R8, R8, R6
       DADDIU R1, R1, -8
       BNE    R1, R2, loop
```

   (a) Identify a stall that could be improved by forwarding and explain in detail how the forwarding case could be detected and handled.

   (b) Explain how a branch delay slot improves performance in general and explain for this loop an instruction that could be moved to the delay slot.
   Now assume the following differences from the 5-stage pipeline:
   - Both the instruction memory and data memory require two cycles (IM1, IM2, DM1, DM2) and
   - the register file is dually ported and there are two write back cycles WBE and WBM so that you can write the result of an execution and a DM reference in the same cycle.

   (c) Show the order of the of the cycles that would mimimize the amount of hardware necessary forwarding.

   (d) List 6 (or all if there are less than 6) forwarding paths; i.e. give the source, the target of the forwarding, and the condition that would cause it to occur.

2. AMAT

(a) Assume an L1 has a hit rate of 90% and the L2 has a hit rate of 80%. Further assume that there is no L3 and references to main memory always hit and we have a block transfer time of 100 cycles from main memory to L2, and the block transfer time from L2 to L1 is 10 cycles. What is the average memory access time? (Assume hit time to L1 is 1 cycle).

(b) Now for the rest of this question assume you break the L1 up into data and instruction caches with hit rates 80% data and 90% instruction. If all instructions and data references are to 1 word quantities and 50% of the instructions are memory reference instructions then

i. What percentage of references to memory are references to instructions memory?

ii. If the What is the AMAT for data references assuming a miss penalty to the unified L2 of 100 cycles? (Note this is not consistent with part a!!!)

iii. What is the overall AMAT?

(c) What is in the TLB?

(d) Describe in detail the process of translating a Virtual address into a physical address.
Is this process implemented in hardware, software or both?

(e) If Pages are 4096 bytes long and the address space is 4GB then how many entries are in the page table?

(f) Given the code below and a direct mapped cache with 256 lines of 32B bytes.

```
double a[8192];
double sum = 0.0;
for(i=0; i < 8192; i=i+4)
      sum = sum + a[i];
```

Assume that the non-array variables are stored in registers and ignore instruction references. What is the hit ratio?

3. Reorder Buffer

   (a) Explain the contents of the ROB.

   (b) Explain a situation that would cause a structural hazard with respect to the CDB.

   (c) What additional hardware is required for a dual commit ROB?

   (d) ROB trace - dual issue
   Assume the following:

   - **Ten ROB slots.**
   - 1 integer Execution unit, 3 reservation stations, one cycle to execute.
   - 1 Floating Add Unit, 3 reservation stations, 6 cycles to execute.
   - 1 Floating MULT Unit, 2 reservation stations, 10 cycles to execute.
   - **Functional units are pipelined.**
   - There is no forwarding between functional units; results are communicated by the common data bus (CDB).
   - The execution stage (EX) does both the effective address calculation and the memory access for loads and stores.
   - Thus, the pipeline is IF/ ID/ IS/ EX/ WB. Loads require one clock cycle.
   - The issue (IS) and write-back (WB) result stages each require one clock cycle.
   - There are five load buffer slots and five store buffer slots.
   - Assume that the Branch on Not Equal to Zero (BNEZ) instruction requires one clock cycle.

```
loop:  LD     F0, 0(R1)
       MULT.D  F8, F0, F0
       ADD.D  F2, F2, F8
       DADDIU R1, R1, +8
       BNE    R1, R2, loop
```

   Fill in the table below for as much as you can assuming the BNE is predicted taken and succeeds. (Do not add rows to the table.)

Iter.	Instruction	Issues at	Execute/memory	Write CDB	Commit

Compilers

1. Consider the following grammar:

$$\begin{array}{rcl} S & \to & V \ = \ E \\ E & \to & E \ - \ E \ \mid \ (\ E \ ) \\ E & \to & V \\ V & \to & \langle id \rangle \ [\ \langle Elist \rangle \ ] \\ \langle Elist \rangle & \to & \langle Elist \rangle \ \mathbf{c} \ E \\ \langle Elist \rangle & \to & \varepsilon \end{array}$$

   (The **c** stands for "comma," and ε denotes the empty string.)

   (a) Construct the sets of LR(1) items for this grammar. (Stop at 8 sets.)
      i. Compute I_0
      ii. For which symbols x is GOTO(I_0, x) nontrivial?
      iii. For each such symbol x, compute GOTO(I_0, x).
      iv. Compute GOTO$([P, \$], =)$, where P is $S \to V \cdot \ = \ E$.

   (b) For the sets of LR(1) items from above, provide entries of the LR(1) parse table.

2. Some programming languages allow the following as a conditional statement:

$$\langle statement \rangle \ ::= \ \mathbf{do} \ \langle statement \rangle_1 \ \mathbf{if} \ \langle test \rangle$$

   where $\langle test \rangle$ is a Boolean-valued expression. This construction is not a loop. It is semantically equivalent to the usual if-statement, "**if** $\langle test \rangle$ **then** $\langle statement \rangle_1$", that is, at runtime, the test is evaluated first, and the statement is executed if and only if the test is true. (There is no "else" clause.)

   Add semantic actions to the production above to translate it into 3-address code or assembly code. Assume that $\langle statement \rangle_1$ and $\langle test \rangle$ are translated into code as they are parsed, and that the result of evaluating $\langle test \rangle$ is stored in the global variable `result`, and so you can say, for example, "`if result then goto` L" immediately after, where L is some label. You may also assume that any intermediate actions you add do not cause any parser conflicts.

   You are allowed the usual two primitives:

   new_label() returns a fresh label to be stored internally

   emit($\cdots$**)** emits its argument to the output code

   You may define any attributes you find helpful, but you may not store data anywhere else (e.g., global variables) besides in attributes of grammar symbols.

   Note that $\langle statement \rangle_1$ emits output code *promptly*; you cannot save the output to be emitted later.

3. The following fragment of intermediate code was generated by a compiler:

```
 1    LabelA: i = 1000
 2            sum = 0
 3    LabelB: j = 1000
 4            k = 1000
 5    LabelC: t1 = j * 4000
 6            t3 = t1
 7            t2 = i * 4
 8            t3 = t1 + t2
 9            t4 = a[t3]
10            if t4 < 0 goto LabelE
11            sum = sum + t4
12            goto LabelF
13    LabelD: k = k - 1
14            if k > 0 goto LabelC
15    LabelE: sum = sum - t4
16    LabelF: j = j - 1
17            if j > 0 goto LabelC
18            i = i - 1
19            if i > 0 goto LabelB
20    LabelG: no op
```

Assume that the only entry point is at `LabelA`.

(a) Decompose the code into basic blocks.

(b) Draw the control flow graph and identify any unreachable code if there is any.

(c) Describe the liveness of each variable just before each line in the block containing statement 10 assuming that only `i`, `j`, `k`, `t4`, and `sum` are live after 10.

(d) Describe any simplifying transformations that can be performed on the code (i.e., transformations that preserve the semantics but improve the code).

Algorithms

1. Find tight asymptotic bounds on any positive-valued function $T(n)$ satisfying the following recurrence for all positive integers n:

$$T(n) = 2T(2n/3) + 3T(n/2) + n^3 \ .$$

   That is, find an expression $f(n)$, as simple as possible, such that $T(n) = \Theta(f(n))$. Show your work.

   (Note: implicit floors or ceilings in the recurrence do not affect the answer.)

2. You are given a standard binary search tree T (with no parent pointers) and two key values $k < \ell$ (which may or may not be in T). Describe an algorithm to list all the keys in T that are *strictly between* k and ℓ. For full credit, your algorithm should take time $O(d + n)$, where d is the depth of T and n is the number of keys that are output (assuming key comparisons take constant time).

3. Let $G = (V, E)$ be a directed acyclic graph with edge weight function $w : E \to \mathbb{N}$ giving each edge $e \in E$ a nonnegative integer weight $w(e)$. Describe an $O(|V| + |E|)$-time algorithm MaxPath(G) that takes G, represented as an adjacency list with vertex array $V[1 \ldots n]$, and outputs the maximum weight of any path from $V[1]$ to $V[n]$, or -1 if no such path exists. Describe informally how you would augment your algorithm to actually compute a maximum-weight path.

   You may define any vertex or edge attributes you find helpful. You may also assume that G is already given in topologically sorted order. That is, for any legal indices i and j, there can be an edge from $V[i]$ to $V[j]$ only if $i < j$.

Theory

Not given in Fall 2015.