CSE Qualifying Exam, Spring 2022

CSCE 513-Computer Architecture

1. MIPS-A and MIPS-B processors have the same Instruction set architecture. The MIPS-A processor has a clock cycle time of 400 picoseconds (ps) and a base CPI of 2.0 for a specific image processing program that includes 20% load and store instructions. Meanwhile, the MIPS-B processor has a clock frequency of 2GHz and the base CPI of 1.2 for the same workload using compliers that generate identical machine code. If the memory hierarchy for both processors have the following characteristics:

- L1 hit time = 0 s, L1 Instruction cache miss rate = 4%, L1D miss rate = 2%
- L2 hit time = 4 ns, L2 miss rate = 1%
- Main Memory access time = 100 ns

(a) What is the effective CPI for MIPS-A and MIPS-B?
(b) Which one of the processors is faster in executing the program and by how much? State any assumptions you make.

2. A computer system with a memory bandwidth of 4.2 GB/s and a peak computational throughput of 5 GFlops/s. Assume:

- The data type of the `res`, `mat`, and `vec` arrays are float (single precision floating point)
- 60% of the arrays fit in on-chip caches cache, while the other 40% should be accessed from the main memory.

What is the expected performance of this computer in GFlops/s when hosting the loop below?

```
for (int i=0;i<100;i++){
      res[i]+=1.0;
      for (int j=0;j<10;j++)
            res[i]+= mat[i][j]*vec[j];
}
```

3. Given the below latencies for the Floating-Point (FP) and Integer operations in a RISC-V processor:

Instruction Producing Results	Instruction Using Results	Latency in clock cycles
FP ALU operation	FP ALU operation	4
FP ALU operation	Store double	3
Load double	FP ALU operation	2
Load double	Store/Load Double	0
Load integer	Integer ALU operation	1
Integer ALU operation	Integer ALU operation	0

(a) What is the IPC (instructions per cycle) of the processor when executing the below program without any scheduling and loop-unrolling?

```
Loop: l.d    $f0,0($s1)      #FP Load
      add.d  $f1,$f0,$f2     #FP Add
      s.d    $f1,0($s1)      #FP Store
      addi   $s1,s1,-8       #integer add
      bne    $s1,$zero,Loop  #Branch
```

(b) What is the maximum IPC that can be achieved when unrolling the above loop with a factor of 3 and use register-renaming and scheduling to speed up the execution?

Spring 2022 CSE Qualifying Exam
CSCE 531, Compilers

1. **Predictive (LL(1)) Parsing**

   Consider the following ("micro-English") grammar, where terminal symbols are in **bold** and the productions are labeled for convenience. Note the bold period (**.**) at the end of the first production. If you need to distinguish terminals from nonterminals in your answer, please underline the terminals.

(1)	S	::=	$Sub\ V\ Obj$ **.**
(2a, 2b, 2c)	Sub	::=	**I** | **a** $Noun$ | **the** $Noun$
(3a, 3b, 3c)	Obj	::=	**me** | **a** $Noun$ | **the** $Noun$
(4a, 4b, 4c)	$Noun$	::=	**cat** | **mat** | **rat**
(5a, 5b, 5c)	V	::=	**like** | **is** | **see** | **sees**

   You are asked to build an LL(1) parser for the micro-English grammar by following a sequence of steps. Since the grammar is very simple, some of the steps will require no work or very little work; still, indicate the result of each step, even if just by writing, "no change." Please clearly label your work according to step number.

   (a) Eliminate left-recursion in the grammar.

   (b) Left-factorize.

   (c) Calculate *Nullable* for each nonterminal.

   (d) Calculate *Nullable* for each production.

   (e) Caclulate *FIRST* for every nonterminal, by using set equations.

   (f) Calculate *FIRST* for each production. (Hint: are any of the nonterminals nullable?)

   (g) Calculate *FOLLOW* for each nonterminal. (As usual, add a new start production before doing this.)

   (h) Make an LL(1) parse table for the micro-English grammar. Use the following order of nonterminals in the table: S', S, Sub, Obj, Noun, V. Use the following order of terminals in the table: **I**, **a**, **the**, **me**, **cat**, **mat**, **rat**, **like**, **is**, **see**, **sees**, **.**, **$**.

   (i) Show the input and stack during table-driven parsing of the string **I see a cat.**.

2. **Syntax-Directed Translation**

Consider the following BNF grammar for statements in a modified fragment of the C language:

$$
\begin{array}{rcl}
\langle \textit{start} \rangle & \rightarrow & \langle \textit{statement} \rangle \\
\langle \textit{statement} \rangle & \rightarrow & \textbf{expr} \ ; \\
& | & \textbf{if} \ (\textbf{expr} \ ) \ \langle \textit{statement} \rangle_1 \ \textbf{else} \ \langle \textit{statement} \rangle_2 \\
& | & \textbf{while} \ (\textbf{expr} \ ) \ \langle \textit{statement} \rangle_1 \\
& | & \{ \ \langle \textit{stmtlist} \rangle \ \} \\
& | & \textbf{break} \ \ \textbf{intconst} \ ; \\
\langle \textit{stmtlist} \rangle & \rightarrow & \texttt{/* null derive */} \\
& | & \langle \textit{stmtlist} \rangle_1 \ \langle \textit{statement} \rangle
\end{array}
$$

Here, the start symbol is $\langle \textit{start} \rangle$. The intended meaning of the `break` n statement (where n is an integer constant) is to break out of n enclosing `while` loops at once. For example,

```
{
    while (1) {
        while (1)
            if (x) break 1; /* goes to x=3, same as usual break */
            else break 2;   /* goes to x=4 */
        x = 3;
    }
    x = 4;
}
```

If $n < 0$ or n is larger than the number of enclosing `while` loops, then this is an error. (If $n = 0$, then the statement has no effect (a no-op); if $n = 1$, then it behaves like a normal break statement would.)

Add semantic actions to the grammar above to handle the generalized break statement (you need not worry about anything else). You may define any attributes you find useful. Assume that attribute **intconst**.val is the numerical value of the **intconst** token. You may assume the predefined functions

getNewLabel() returns a fresh label,

emitLabel(label) writes the label to the output stream, followed by a colon,

emitJump(label) writes an unconditional jump instruction to the given label, and

error(msg) signals an error and gives msg as an explanation.

3. **Intermediate Code Analysis/Optimization**

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.

(a) (20% credit) Decompose the code into basic blocks B1,B2, ..., giving a range of line numbers for each.

(b) (20% credit) Draw the control flow graph, and describe any unreachable code.

(c) (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	...

(d) (20% credit) Describe any simplifying tranformations 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).

Spring 2022 CSE Qualifying Exam
CSCE 551, Theory

1. (Fenner, Spring 2022) Fix an alphabet Σ. For string $w \in \Sigma^*$ and language $L \subseteq \Sigma^*$, we define the *insertion* operation $w \to L$ to be the language of all possible strings obtained by inserting w into a string in L. (Formally, $w \to L := \{ywz : yz \in L\}$.) For example,

$$bb \to \{aaa\} = \{bbaaa, abbaa, aabba, aaabb\} .$$

   (a) (80% credit) Show that if L is regular and w is any string, then $w \to L$ is regular.

   (b) (20% credit) For languages $L_1, L_2 \subseteq \Sigma^*$, define

$$L_1 \to L_2 := \bigcup_{w \in L_1} w \to L_2 .$$

   Show that if L_1 and L_2 are regular, then $L_1 \to L_2$ is regular.

2. (Fenner, Spring 2022) Let Σ be an alphabet and let $L \subseteq \Sigma^*$ be a language such that

   - L is Turing-recognizable, and
   - for every integer $n \geq 0$, there exist exactly n many strings in L of length $\leq 2^n$, that is,

$$(\forall n \geq 0) \ |\{w \in L : |w| \leq 2^n\}| = n .$$

   Show that L is decidable.

3. (Fenner, Spring 2022) The DISJOINT CIRCUITS problem is

   Instance: A graph G.
   Question: Are there two circuits in G such that every vertex of G lies on exactly one of the two circuits?

   Show that DISJOINT CIRCUITS is NP-complete. To show NP-hardness, polynomially reduce from HAMILTONIAN CIRCUIT.

Spring 2022 Q-exam — CSCE 750 (Algorithms)

1. **(Solving a Recurrence)**

   Let $T(n)$ be any positive-valued function defined for all integers $n \geq 1$ by the following recurrence:

   $$T(n) = T(n^{1/3}) + T(n^{2/3}) + \lg n \; .$$

   Find an expression $f(n)$, as simple as possible, such that $T(n) = \Theta(f(n))$. Use the substitution method to **prove** that your answer is correct. You may assume that any implicit floors and ceilings are of no consequence.

2. **Optimal vacation time**

   Finley is starting a new temporary job as a research intern at a large company. He is interested in maximizing the amount of fun he can have away from work by carefully selecting vacation days, i.e. days on which he chooses not to work. This problem asks for an efficient algorithm to help Finley determine an optimal selection of vacation days.

   Finley numbers the days of the internship, which lasts for n total days, with integers $0, 1, \ldots, n-1$ and estimates the *fun value* $F[i]$ of taking vacation on the i^{th} day. His goal is to select a set of vacation days $V \subseteq \{0, 1, \ldots, n-1\}$ to *maximize the total fun value*. That is, Finley would like to choose V to maximize $\sum_{i \in V} F[i]$.

   However, there is one complication: Though Finley's internship does not limit the number of vacation days he may take, company policy prevents any pair of vacation days from being fewer than m days apart. That is, if Finley takes vacation on day i, the next possible vacation day is on day $i + m$.

   Describe a $\Theta(n)$-time algorithm whose inputs are an array $F[0, \ldots, n-1]$ of integers, representing the fun value of each possible vacation day, and a positive integer m, representing the minimum spacing between successive vacation days, and whose output is the *the maximum total fun* that can be achieved while satisfying the minimum spacing constraint. (You do not need to worry about computing a specific set of vacation days that achieves this maximum total fun.) **Explain** why your algorithm works, in enough detail to convince an intelligent but skeptical reader that it is correct.

3. **Verifying Correctness**

   Prof. Proper claims to have implemented Dijkstra's algorithm to find single-source shortest paths in a digraph, and your job is to check that her algorithm runs correctly on a given input. She provides you with the digraph $G = (V, E)$ in adjacency list representation, with edge weights $w(e) \geq 0$ for every $e \in E$, along with the source vertex $s \in V$. Each vertex $v \in V$ also has a d-value and a π-value computed by her algorithm. All d-values are real numbers, and all π-values are vertices in V (except $s.\pi$, which is NIL).

   You should assume that all vertices of G are reachable from s.

   (a) (60% credit)

   Describe an algorithm that decides whether the set of edges

   $$T := \{(v, v.\pi) : v \in V,\ v \neq s\}$$

form a single tree with root s and child-to-parent edges (as they should after Dijkstra's algorithm). Here, the edge weights and d-values are irrelevant. Your algorithm should run in time $O(|V|)$. **Explain** your algorithm in enough detail so that an intelligent programmer can implement it.

(b) (Full credit)

Describe an algorithm that takes all the data given above and checks whether the d- and π-values reflect a correct implementation of Dijkstra's algorithm with source vertex s. Unfortunately, your algorithm must run in time $O(|V| + |E|)$, so you don't have enough time to run Dijkstra's algorithm yourself and compare your results with hers. **Explain** your algorithm in enough detail so that an intelligent programmer can implement it.

If you want, you can assume that $V = \{v_1, \ldots, v_n\}$, $s = v_1$, and shortest paths are all unique (no ties).