Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 1996

October 15, 1996

PLEASE READ THIS FIRST

- You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.
- Be sure you have all the pages of this exam. There are three pages in addition to this.
- This exam is OPEN BOOK. You may use notes, articles, or books — but no help from people or computers.
- Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea does not work out. You can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when we think it is false.
- Points in this exam add up to 100.

1 Blind Search (20 points)

Here is the description of breadth-first search from Ginsberg's book (page 24):

1. Set L to be a list of the initial nodes in the problem.
2. Let n be the first node on L. If L is empty, fail.
3. If n is a goal node, stop and return it and the path from the initial node to n.
4. Otherwise, remove n from L and add to the end of L all of n's children, labelling each with its path from the initial node. Return to step 2.

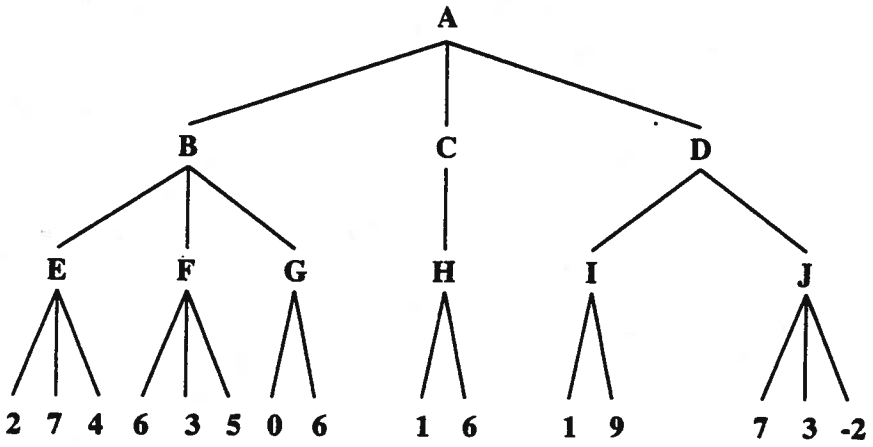
Once this algorithm generates a goal node, the length of the path from the initial state to the goal state will be as short as possible.

This property holds under the assumption that all arcs (edges) in the tree have the same length (or cost). Modify breadth-first search to ensure that the above property holds even when arc lengths are not necessarily equal (but the lengths are known).

Just describe the changes, do not rewrite the algorithm.

2 Adversary Search (20 points)

Consider the following Minimax tree. The first player is the maximizer.



a. What is the solution? That is, which move should be made next and what is the expected value of that move?

b. Using the alpha-beta pruning method (and standard left-to-right evaluation of nodes), how many leaves of the tree get evaluated?

c. Using alpha-beta pruning, but right-to-left evaluation of nodes, how many leaves get evaluated?

(9)

3 Learning (20 points)

You are in charge of machine induction of classifiers at the Grand Kernel Bank. A request comes in from the financial officer Dr. Penny De L'Argent to induce a classifier from data on fourteen cases that concern the creditworthiness of applicants for a special Gold Credit Card, which grants a high credit limit and other privileges. The classifier should determine whether an applicant is creditworthy. Note that the Bank also grants regular cards, and some applicants for the Gold Card already hold the regular card. Past applicants who have been granted the Gold card are to be considered creditworthy.

Case	high income?	already holds regular card?	was granted gold card?
1	yes	yes	yes
2	yes	yes	yes
3	yes	yes	yes
4	yes	yes	no
5	yes	no	no
6	yes	no	yes
7	yes	no	no
8	no	no	no
9	no	no	no
10	no	no	no
11	no	no	no
12	no	no	yes
13	no	no	yes
14	no	yes	yes

What tree, including classifications, would be output by an ID3-style discrimination tree induction algorithm?

[Hints: the last column in the training table is used to determine whether a given training instance is positive or negative. This leaves two feature columns. You may want to number these 0 and 1. To answer this question, it is not necessary to compute the formula for information gain, but just to understand what it means.]

4 Probability (20 points)

Males are 47 percent of the population; 30 percent of all males smoke, and 25 percent of the entire population smokes. What fraction of all smokers are female? You may assume that people are either male or female. A numerical formula is sufficient.

(10)

5 Vision (20 points)

Ginsberg (page 329) draws a distinction between these two types of edges:

1. edges in the image, and

2. edges in the objects the image depicts.

Edges of type 1 are abrupt changes in image intensity, while edges of type 2 are discontinuities in the depth or in the tangent of a surface in the world.

Ginsberg also mentions image noise as one reason why the two types of edges may not coincide. Argue, by example or otherwise, that there are more fundamental reasons why these two types of edges do not in general coincide.

(11)

Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 1996

Solution Samples

1 Blind Search

Breadth-first with variable costs (or lengths) is called Uniform Cost Search. All we need to change is to attach to each node in L a variable g with the cost of the path from the root to the node itself. Then, we choose from L the node with the smallest g, rather than the first node of the list.

2 Adversary Search

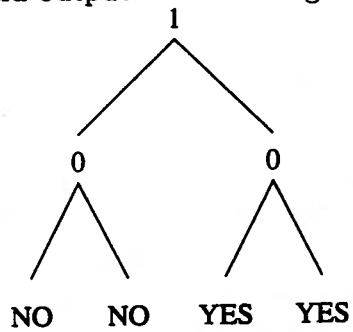
a. You should move to node D, whose value is 7.

b. All 15 will be evaluated if you go left-to-right.

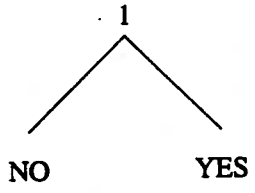
c. Going right-to-left, 8 leaves will be evaluated: all three leaves under J, the 9 under I, both under H, and both under G.

3 Learning

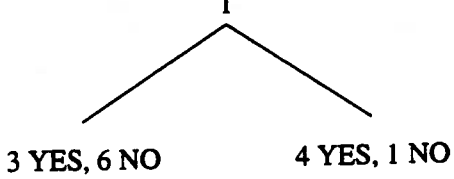
ID-3 would output the following tree:



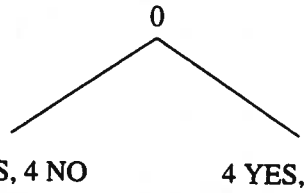
or equivalently



The important thing to notice about this tree is that feature 1 is at the root. The reason for this choice is that feature 1 splits the training set as follows:



while feature 0 splits it as follows:



This shows feature 1 to yield the highest information gain. In fact, feature 0 does next to nothing to the balance of positive and negative features in the training set, while feature 1 splits the training set into mostly negative and mostly positive instances.

4 Probability

Let A be the event "a person smokes", and B be the event "a person is male". We can then define the following probabilities or fractions:

$$\begin{aligned} P(A) &= \text{fraction of smokers in the entire population} = 0.25 \\ P(B) &= \text{fraction of males in the population} = 0.47 \\ P(A|B) &= \text{fraction of smokers in the male population} = 0.30\,. \end{aligned}$$

The required probability is then given by Bayes' theorem:

$$\begin{aligned} 1 - P(B|A) &= \text{fraction of females among all smokers} \\ &= 1 - \frac{P(A|B)P(B)}{P(A)} = 1 - \frac{0.30 \times 0.47}{0.25} = 1 - 0.564 = 0.436\,. \end{aligned}$$

Thus, 43.6 percent of smokers are female.

5 Vision

Edges of type 1 are photometric events, that is, they relate to how light bounces off different surfaces into the camera. Edges of type 2 are geometric events, that is, they relate to the shape of objects. The fact that these are completely different phenomena should already raise a flag: why should edges of the two types coincide at all? Examples of edges of type 1 (photometric) that correspond to no edge of type 2 abound: any surface marking (text on an object, change in paint color, etc) generates a photometric edge, but corresponds to no surface or tangent discontinuity. The converse (geometric edges without photometric edges) occurs quite often when the foreground and the background happen to have the same brightness in the image. Chameleons in the jungle and polar bears in snowy environments rely on this lack of coincidence for their survival.

Computer Science Department
Stanford University
Comprehensive Examination in Automata and Formal Languages
Autumn 1996

October 14, 1996

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in BLUE BOOKS. There are three problems in the exam. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. The number of POINTS for each problem indicates how elaborate an answer is expected. The exam takes 30 minutes.

3. This exam is OPEN BOOK. You may use notes, articles, or books—but no help from other sentient agents such as other humans or robots.

4. Show your work, since PARTIAL CREDIT will be given for incomplete answers.

Automata and Formal Languages (30 points)—Autumn 1996

1. [10] Give context-free grammars generating the following languages over the alphabet $\{a, b\}$. (a) $\{a^i b^j a^{i+j+k} b^k \mid i, j, k \geq 0\}$; (b) All strings with an equal number of a's and b's.

2. [10] Consider the following decision problem. Given a regular expression over the alphabet $\{a, b\}$, decide whether the language it denotes contains a string of palindromes. (A string of palindromes is a member of P^* where P is the set of all palindromes.) State whether this problem is decidable, and briefly sketch the reason.

3. [10] For each of the following sets, state whether it is in NP, coNP, or neither. (If it is both you need not say so, in which case either NP or coNP is a correct answer.) Give a one-sentence reason for each.

(a) The set of three-colorable undirected planar graphs. (A three-coloring of a graph is an assignment of at most three distinct colors to the vertices of the graph such that every edge has different colors at each end.)

(b) The set of finite sets of signed binary integers having no subset adding to zero.

(c) (The knapsack problem). The set of pairs (w, c) where w is a finite list of positive binary numbers (items available to go in the knapsack) and c is an integer (the size of the knapsack) such that some subset of the elements of w sum to c (a perfect packing of the knapsack).

(d) The set of pairs (i, j) of nonnegative integers written in binary such that $\varphi(i, j)$ holds but $\varphi(i', j)$ does not hold for $0 \leq i' < i$, where the predicate φ is computable in time polynomial in the length of its inputs written in binary.

(e) The set of valid computations of a universal Turing machine.

(15)

Comprehensive Exam Solutions
Automata and Formal Languages (30 points)
Autumn 1996

1. [10] Give context-free grammars generating the following languages over the alphabet $\{a, b\}$. (a) $\{a^i b^j a^{i+j+k} b^k \mid i, j, k \geq 0\}$;

$$
\begin{aligned}
S &\rightarrow AC && (1)\\
A &\rightarrow aAa && (2)\\
A &\rightarrow B && (3)\\
B &\rightarrow bBa && (4)\\
B &\rightarrow \epsilon && (5)\\
C &\rightarrow aCb && (6)\\
C &\rightarrow \epsilon && (7)
\end{aligned}
$$

(b) All strings with an equal number of a's and b's.

$$
\begin{aligned}
S &\rightarrow aSbS && (8)\\
S &\rightarrow bSaS && (9)\\
S &\rightarrow \epsilon && (10)
\end{aligned}
$$

2. [10] Consider the following decision problem. Given a regular expression over the alphabet $\{a, b\}$, decide whether the language it denotes contains a string of palindromes. (A string of palindromes is a member of P^* where P is the set of all palindromes.) State whether this problem is decidable, and briefly sketch the reason.

Decidable. The problem is equivalent to whether the given regular expression has a nonempty intersection with the set of strings of palindromes,

(16)

a context-free language. Given a regular language and a context-free language, their intersection is context-free and one may obtain a context-free grammar generating it and then test emptiness of the language so generated.

3. [10] For each of the following sets, state whether it is in NP, coNP, or neither. (If it is both you need not say so, in which case either NP or coNP is a correct answer.) Give a one-sentence reason for each.

(a) The set of three-colorable undirected planar graphs. (A three-coloring of a graph is an assignment of at most three distinct colors to the vertices of the graph such that every edge has different colors at each end.)

In NP, with membership witnessed by a planar embedding and a 3-coloring, checkable in linear time.

(b) The set of finite sets of signed binary integers having no subset adding to zero.

In coNP, with nonmembership witnessed by a subset adding to zero, checkable in linear time.

(c) (The knapsack problem). The set of pairs (w, c) where w is a finite list of positive binary numbers (items available to go in the knapsack) and c is an integer (the size of the knapsack) such that some subset of the elements of w sum to c (a perfect packing of the knapsack).

In NP, with membership witnessed by a subset of w summing to c, checkable in linear time.

(d) The set of pairs (i, j) of nonnegative integers written in binary such that $\varphi(i, j)$ holds but $\varphi(i', j)$ does not hold for $0 \leq i' < i$, where the predicate φ is computable in time polynomial in the length of its inputs written in binary.

In coNP, with nonmembership witnessed by i, i', j such that $i' < i$ and at least one of $\varphi(i, j)$ or $\varphi(i', j)$ is false.

(e) The set of valid computations of a universal Turing machine.

In P, hence in both NP and coNP.

Computer Science Department
Stanford University
Comprehensive Examination in Compilers
Autumn 1996

October 17, 1996

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. Be sure you have all the pages of this exam. There is 1 page.

   The exam takes 30 minutes.

3. This exam is OPEN BOOK. You may use notes, articles, or books—but no help from other sentient agents such as other humans or robots.

4. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

CS Comprehensive Exam: Compilers (30 points)

Question 1 (9 points)

Consider the following block of 3-address code:

```
1)  a := b + c
2)  d := b
3)  e := c + d
4)  f := a - e
5)  d := a
```

Assume that the value of each of a through f is needed following the block. Give six transformations that may be performed on this block to improve the running time on a typical machine. Indicate the change made at each of your six steps (it is sufficient to give the rewritten line only, rather than the whole block) and *give the name of the transformation or brief justification.* You may apply more than one transformation to a single statement, but use separate steps to do so.

Question 2 (11 points)

For the grammar

$$\begin{array}{lrcl} 1) & S & \rightarrow & aS \\ 2) & S & \rightarrow & b \end{array}$$

a) [8 points] Construct the LR(0) sets of items (sometimes called SLR items; they involve no lookahead) and show their transition diagram.

b) [3 points] Give the LR(0) parsing table and indicate any conflicts that arise.

Question 3 (10 points)

Suppose we wish to build a lexical analyzer that recognizes the following three tokens:

1) **01**
2) **010**
3) **0*10***

Assuming the LEX priority rules (longest token recognized, and in case of ties, pick the token listed first), draw a deterministic finite automaton that recognizes the above three tokens. Indicate, for each state, which, if any, token is recognized. Also, since a token may not be recognized until further symbols are read by the automaton, indicate how far back the recognized token ends. For example, attaching $(2, 3)$ to an accepting state means that the second token, **010**, was recognized, and the last 3 symbols read by the automaton are not part of the token.

(19)

Solutions to 1996 CS Comprehensive Exam: Compilers

Question 1 (9 points)

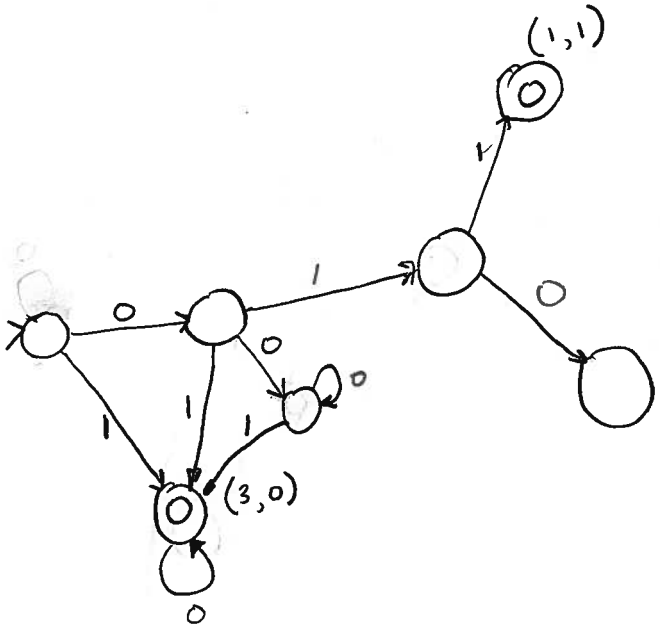
Consider the following block of 3-address code:

```
1)  a := b + c
2)  d := b
3)  e := c + d
4)  f := a - e
5)  d := a
```

Assume that the value of each of a through f is needed following the block. Give six transformations that may be performed on this block to improve the running time on a typical machine. Indicate the change made at each of your six steps (it is sufficient to give the rewritten line only, rather than the whole block) and *give the name of the transformation or brief justification.* You may apply more than one transformation to a single statement, but use separate steps to do so.

Solution

1. Replace line (3) by `e := c+b` [copy propagation, using line (2)].
2. Replace line (3) by `e := b+c` [use of commutative law].
3. Replace line (3) by `e := a` [common subexpression elimination, using line (1)].
4. Replace line (4) by `f := a-a` [copy propagation, using line (3)].
5. Replace line (4) by `f := 0` [algebraic simplification].
6. Eliminate line (2) [dead code elimination].



(20)

Question 2 (11 points)

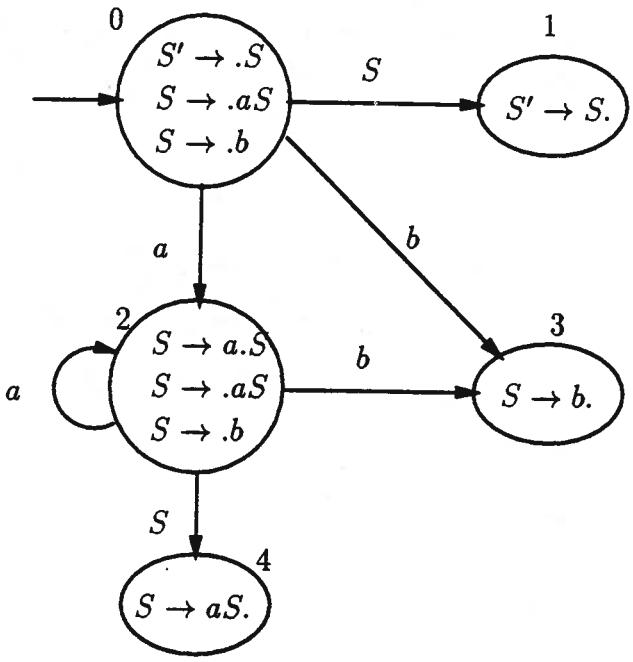
For the grammar

$$\begin{array}{lllll} 1) & S & \rightarrow & aS \\ 2) & S & \rightarrow & b \end{array}$$

a) [8 points] Construct the LR(0) sets of items (sometimes called SLR items; they involve no lookahead) and show their transition diagram.

b) [3 points] Give the LR(0) parsing table and indicate any conflicts that arise.

Solution (a)



Solution (b)

State	Action			Goto
	a	*b*	$	*S*
0	shift 2	shift 3		1
1			accept	
2	shift 2	shift 3		4
3			reduce $S \rightarrow b$	
4			reduce $S \rightarrow aS$	

(21)

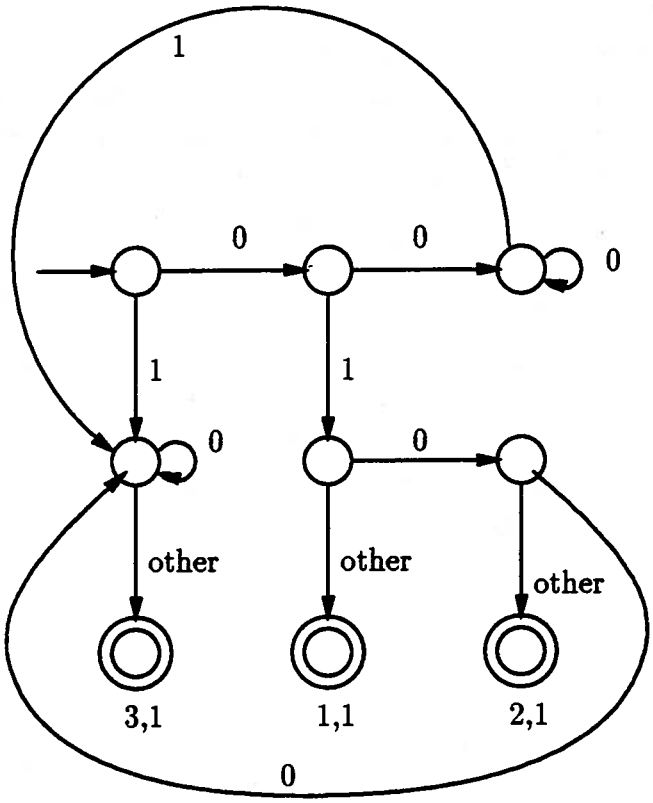
Question 3 (10 points)

Suppose we wish to build a lexical analyzer that recognizes the following three tokens:

1) **01**
2) **010**
3) **0*10***

Assuming the LEX priority rules (longest token recognized, and in case of ties, pick the token listed first), draw a deterministic finite automaton that recognizes the above three tokens. Indicate, for each state, which, if any, token is recognized. Also, since a token may not be recognized until further symbols are read by the automaton, indicate how far back the recognized token ends. For example, attaching $(2, 3)$ to an accepting state means that the second token, **010**, was recognized, and the last 3 symbols read by the automaton are not part of the token.

Solution



(22)

Test only • No solutions available

Computer Science Department
Stanford University
Comprehensive Examination in Computer Architecture
Autumn 1996

October 16, 1996

READ THIS FIRST!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. This exam is CLOSED BOOK. You may not use notes, articles, or books.

3. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out; you can also get credit for realizing that certain approaches are incorrect. On a true/false question, you might get partial credit for explaining why you think something is true when it is actually false. But no partial credit can be given if you write nothing.

TEST IS 7 pages. 1 HOUR long.

Computer Science Comprehensive Examination
Computer Architecture
(60 points)

Question 1 (15 points)

Early next year, Intel will add an extension to the x86 ISA aimed at improving performance of multimedia oriented programs on its Pentium™ processor chips. These instructions, collectively called the MMX extensions, can optimize execution of ALU instructions that operate on 8 / 16 / 32 bit signed or unsigned integers by "packing" several of these smaller integers into special 64-bit registers and then performing the same ALU operations on these integers in parallel with a single instruction.

We are interested in determining the effectiveness of this extension. A suite of benchmark programs has been selected with the following instruction counts and CPIs on a non-pipelined machine:

	# of instructions	CPI
Integer ALU		
(8-bit)	10,000	4
(16-bit)	20,000	4
(32-bit)	40,000	4
(64-bit)	10,000	4
Load	15,000	5
Store	15,000	4
Branches	5,000	3

After examination of the 8-bit and 16-bit integer ALU operations, we discover the following:

	8-bit ALU	16-bit ALU	32-bit ALU
% can be optimized	80%	70%	20%
average number of useful packed integers / MMX instruction	5	3	2

The CPI's of the MMX instructions are the same as the original ALU instructions. There is also an overhead of loading the integers into the special CPU registers. For this benchmark, an extra 5000 move instructions (CPI = 3) are required to perform the necessary register-register moves.

a. (5) What is the performance improvement on the benchmark with the ISA extension if the clock cycle does not change on the machine with extensions?

(24)

b. (5) If the benchmark is currently running on a 50MHz machine with MMX extensions, what would the new clock frequency on the original machine have to be in order to have the same performance as the machine with the ISA extension?

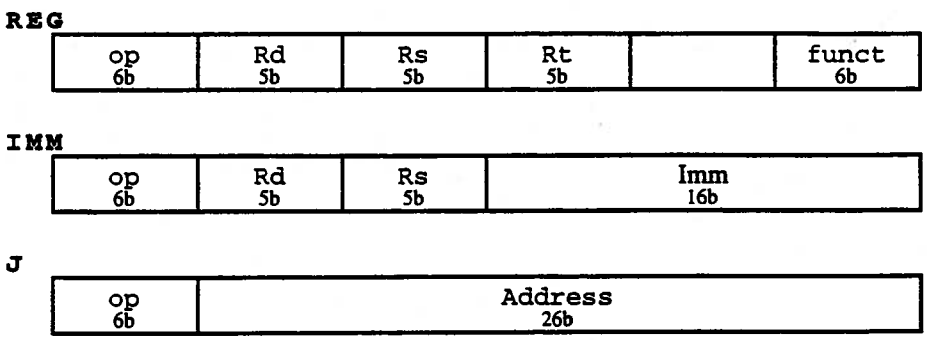
c. (5) What changes to the machine and issues must be considered to accommodate these new instructions? Based on these issues and changes associated with adding these instructions, do you think this is a good idea? Why?

(25)

Question 2 (25 points)

On the next page is a basic single-cycle MIPS processor datapath with some missing information:

- 32-bit byte-aligned PC
- 3 instruction formats:

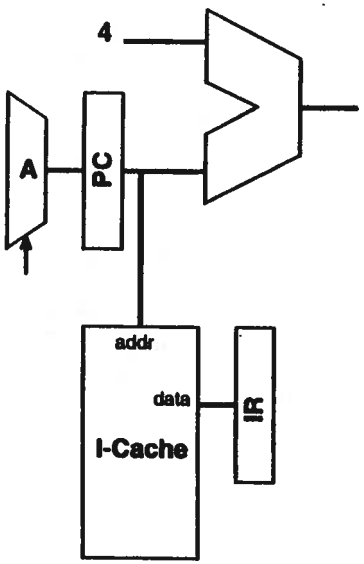


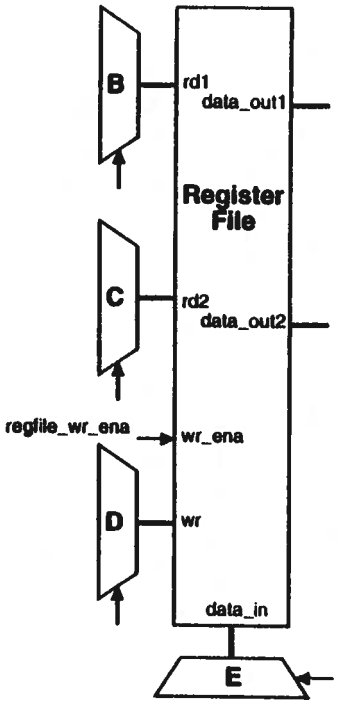
- The ALU can perform unsigned or signed addition or subtraction operations

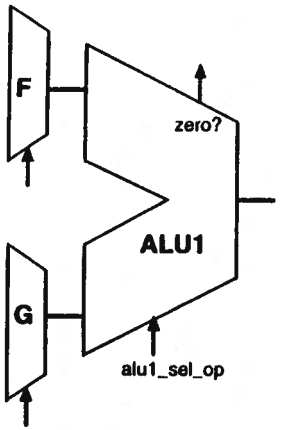
Show the necessary inputs into the multiplexers and the bit widths of the missing wires necessary to support the following list of instructions in the diagram. Also show the correct setting of the multiplexers and control wires for each instruction in the table given on the following page:

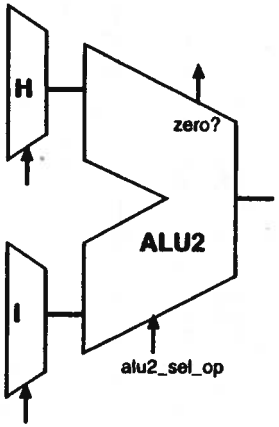
ADDI - Add signed Rs and Imm and place result in Rd
SUBU - Subtract unsigned Rt from Rs and place result in Rd
BEQAL - Branch to PC + Imm (Imm is a signed offset in number of instructions) and place the address of the next instruction after the current PC in R31 if Rs and Rt are equal, otherwise go to the next sequential PC.
J - Jump to the word address in the instruction concatenated with the upper bits of the next instruction after the current PC.
LW - Load the contents of the address at Rs + Imm (Imm is a signed offset in bytes) into Rd
SW - Store Rd to the address at Rs + Imm (Imm is a signed offset in bytes)

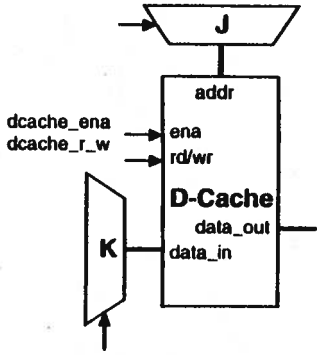
(26)


4
A
PC
addr
data
I-Cache
IR


B
rd1
data_out1
Register File
C
rd2
data_out2
regfile_wr_ena
wr_ena
D
wr
data_in
E


F
zero?
ALU1
G
alu1_sel_op


H
zero?
ALU2
I
alu2_sel_op


J
addr
dcache_ena
ena
dcache_r_w
rd/wr
D-Cache
data_out
K
data_in

	ADDI	SUBU	BEQAL	J	LW	SW
A						
B						
C						
D						
E						
F						
G						
H						
I						
J						
K						
alu1_sel_op						
alu2_sel_op						
regfile_wr_ena						
dcache_ena						
dcache_r_w						

(25)

Question 3 (20 points)

Shown below are the contents the TLB, page table and cache of the MicroMachines MiniCPU™, a miniaturized version of the modern CPU. The MiniCPU™ has the following characteristics:

- 16-bit virtual address, 15-bit physical address, both byte aligned
- TLB
  - 4 entries, direct mapped
  - 5% miss rate
  - No translation penalty for a hit
  - Each miss requires an average of 30~ (cycles) to process
- Page table
  - 16 entries total to map all possible pages in a process' virtual memory space
  - Page table accesses are accounted for in the TLB miss time
- Cache
  - Write back, write allocate
  - There is a infinitely large write buffer to main memory which can hide all write latencies to main memory
  - 50% of the lines are dirty
  - 16 entry, 2-way set associative
  - Each cache line is 32 bytes long
  - 10% miss rate
  - No penalty on a hit
  - A lookup in the cache returns a word (4 bytes)
- Main memory
  - It first takes 80ns to lookup a random address in main memory. Once this location is found, the bus can transfer data at 50Mbytes/s.
  - Sequential memory access latencies are hidden by the bus transfer time.
  - An entire cache line must be transferred before a miss in the cache can be satisfied.

a. (5) Show the tag, index and offset bits necessary to access the TLB with a virtual address and to access a word in the cache with a physical address:

VIRTUAL

PHYSICAL

b. (5) Suppose we wanted to do the cache and TLB lookup in parallel. What restrictions does this place on the TLB/cache sizes, if any?

c. (5) What is the AMAT (average memory access time) of this memory system if the CPU is running at 25MHz? How much would the AMAT improve if the cache lookup is done in parallel with the TLB lookup (assuming it is possible with the current configuration)?

d. (5) Discuss the tradeoffs between a direct mapped and a highly associative cache (e.g. 4-way set associative with LRU replacement). What sort of reference streams would benefit from an associative cache? If you only had a direct mapped cache available, what could you do to your code and data structures at compile time to improve performance on these reference streams?

(30)

SOLUTIONS

++
==
++
Database Comprehensive Exam, October 18 1996

Answer 1.

Relational Algebra
(a) PROJECT[E1.studentID] (
SELECT[E1.studentID = E2.studentID and
E1.course# = C1.course# and
E2.course# = C2.course# and
C1.dept <> C2.dept] (
(enroll(E1) X enroll(E2) X course(C1) X course(C2))))

where rel-name(R) renames relation rel-name as R.

Answer 2. SQL 1:

```
SELECT DISTINCT studentID
FROM Enroll
WHERE NOT EXIST
        (SELECT *
        FROM Course
        WHERE Enroll.course#=Course.course# and dept="physics")
```

Answer 3. SQL 2:

2.a Best: SELECT B FROM R; (since duplicates may be returned)
2.b Next best: select avg(A) from R; (since avg can't be expressed)

Answers 4.

Answer 4.a: 1. ID or name can be key
Answer 4.b: 2. Need name (or Id) and dept
Answer 4.c: 3. Also need partial crossproduct

Answers 5:

Answer 5.a: AB, BC, and BD.

Answer 5.b: no. All attributes are in some key, so there cannot be a 3NF violation.

Answer 5.c: C->D or D->A are examples of BCNF violations.
In each case the left side does not include a key.

NUMERICAL ANALYSIS

1996-1997

Comprehensive Examination

closed book

1. (15 points) Show that Newton's method for a solution x of $f(x) = 0$ for a nonlinear scalar real valued function converges quadratically under certain stated assumptions on f.

2. (15 points) Derive an expression for the error in polynomial interpolation of a sufficiently smooth real valued function f(x) by a polynomial of degree n. State conditions on f needed to obtain the result. Assume that the error e(x) has the form $e(x) = c(x)(x-x_0)\cdots(x-x_n)$ where the xj are the interpolation points and compute c(x).

Computer Science Department

Stanford University

Comprehensive Examination in Software Systems

Autumn 1996

READ THIS FIRST!!!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

2. The number of POINTS for each problem indicates how elaborate an answer is expected. For example, an essay-type question worth 6 points or less doesn't deserve an extremely detailed answer, even if you feel you could expound at length upon it.

3. The total number of points is 60, and the exam takes 60 minutes. This "coincidence" can help you plan your time.

4. This exam is CLOSED BOOK. You may NOT use notes, books, computers, other people, etc.

5. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea doesn't work out. You can also get credit for realizing that certain approaches are incorrect.

6. If you are convinced you need to make an assumption to answer a question, state your assumptions(s) as well as your answer.

7. Be sure to provide justification for your answers.

1996 Comprehensive Exam: Software Systems (60 points total)

1. (15 points) Solaris 2 uses "adaptive mutexes" to protect access to every critical data item. An adaptive mutex starts as a standard semaphore implemented as a spinlock. If the data are locked (already in use), the adaptive mutex does one of two things. If the lock is held by a thread that is currently running, the thread waits for the lock to become available. If the thread holding the lock is not currently in the run state, the thread blocks, going to sleep until it is awakened by the lock being released.

   a. Why are both options (spinning and sleeping) provided when the data are already locked, and why does the decision depend on whether the thread holding the lock is running?

   b. For what sort of application workload is this feature beneficial?

   c. On a uniprocessor system, how do adaptive mutexes behave?

2. (10 points) Page replacement in a virtual memory system can be either global or local. In global page replacement a page fault is satisfied by selecting a page from the set of all physical pages. In local page replacement, the page fault can only be satisfied from the faulting process' own set of physical pages. The number of physical pages allocated to a process in the local scheme does not change over its lifetime.

   a. What are the potential disadvantages of each scheme?

   b. Why is global replacement more commonly found in existing systems?

3. (10 points) Monitors are a popular mechanism for ensuring synchronization of processes that access system data structures and resources. However, monitors do not guard against starvation.

   a. Briefly describe why monitors do not guard against starvation.

   b. Given this problem, why is it practical to use monitors in real-world systems? (How is the starvation problem handled in practice?)

4. (20 points) File cache update policies can be categorized by the amount of delay they allow before file updates must be written to disk. Compare the following policies based on reliability, process latencies, and disk throughput. For each policy, also describe a workload that would benefit from the specific policy.

   •Write-through immediately (no delayed writes)

   •Write-through on close (write modified data to disk when the file is closed. The close operation doesn't return until the write finishes. If a process with open files exits, the system calls the close operation on these files before the exit completes.)

   •Delay for 30 seconds (write out modified data from the cache when it's 30 seconds old)

   •Delay for 5 minutes (write out modified data from the cache when it's 5 minutes old)

5. (5 points) In UNIX, there are separate system calls to start a new process in the image of its running parent (`fork`) and to load into a process's address space fresh code and data from an executable that it should use for execution (`exec`). Why doesn't UNIX use just one system call that both creates a new process and loads the desired image (code and data)?

1996 Comprehensive Exam: Software Systems (60 points total)

1. (15 points) Solaris 2 uses "adaptive mutexes" to protect access to every critical data item. An adaptive mutex starts as a standard semaphore implemented as a spinlock. If the data are locked (already in use), the adaptive mutex does one of two things. If the lock is held by a thread that is currently running, the thread waits for the lock to become available. If the thread holding the lock is not currently in the run state, the thread blocks, going to sleep until it is awakened by the lock being released.

   a. Why are both options (spinning and sleeping) provided when the data are already locked, and why does the decision depend on whether the thread holding the lock is running?

Spinning makes sense if the lock will soon be released, since you avoid the overhead of a context switch. Sleeping makes sense if the lock will be held for a while, since you don't want to consume CPU resources needlessly for a long time. The state of the thread holding the lock is a hint about how long the lock may be held. If the thread is not running, it might be a while before it releases the lock. The fact that it's not running could indicate that it's waiting for I/O to complete, or some other such long operation.

   b. For what sort of application workload is this feature beneficial?

One example is that a real-time multi-processor workload might benefit from this approach if locks are only held for a short time. Threads are likely to get the locks they need very quickly if the threads holding them are still running. Workloads that demand efficient use of a multiprocessor also benefit, since if a lock is held for a long time, threads waiting for it will get out of the way and make the processor available for another thread to run.

   c. On a uniprocessor system, how do adaptive mutexes behave?

On a uniprocessor, only one thread can run at a time. Thus, the thread already holding the lock can never be running. This means the thread requesting the lock will always sleep, giving up the processor. This makes it easier for the thread holding the lock to be rescheduled and (one hopes) give up the lock soon.

2. (10 points) Page replacement in a virtual memory system can be either global or local. In global page replacement a page fault is satisfied by selecting a page from the set of all physical pages. In local page replacement, the page fault can only be satisfied from the faulting process' own set of physical pages. The number of physical pages allocated to a process in the local scheme does not change over its lifetime.

   a. What are the potential disadvantages of each scheme?

Disadvantages of global scheme: A process cannot control its own paging behavior, since its pages can be taken away by other processes. One heavily-faulting process could hurt the performance of other small, well-behaved processes.

Disadvantages of local scheme: A process that needs more pages cannot get the benefit of free memory that is under-utilized by some other process. The allocation of pages per process is restrictive. A process that changes behavior over its lifetime can't increase the number of pages allotted to it or give up pages if it no longer needs so much memory.

   b. Why is global replacement more commonly found in existing systems?

A process that needs more memory can take advantage of free pages in the global pool, so memory is more efficiently used overall. Thus system throughput is generally higher in the

global scheme.

3. (10 points) Monitors are a popular mechanism for ensuring synchronization of processes that access system data structures and resources. However, monitors do not guard against starvation.

   a. Briefly describe why monitors do not guard against starvation.

A process waiting on the monitor lock may not be the first one to run when the lock is released. If a new process happens to execute the monitor before the other process is woken up, the new process will get the monitor lock first. If new or lucky processes keep testing the monitor lock at just the right time, then the poor waiting process may wait forever. Note that there are also some definitions of monitor semantics where a process waiting on a condition variable may be "signalled" or "notified," only to find by the time it's scheduled to execute that the condition for which it is waiting is no longer true because another process has run just before it and has changed something. The unlucky process will then have to wait on that condition again. This could go on forever.

   b. Given this problem, why is it practical to use monitors in real-world systems? (How is the starvation problem handled in practice?)

In practice we rely on at least several things to make this scenario unlikely. First, a nicely behaving scheduler can help. Second, we try not to run systems with resources so scarce that there are likely to be so many processes wanting them simultaneously. Third, careful design of the system is important to make sure that monitor locks are not held for a long time, and that particular monitors do not become bottlenecks for resources under a lot of demand. For example, we can balance locking around whole sub-systems with finer-grained locking for individual items. One lock around the whole file system could be a bad idea. A lock per file-data-structure may make more sense.

4. (20 points) File cache update policies can be categorized by the amount of delay they allow before file updates must be written to disk. Compare the following policies based on reliability, process latencies, and disk throughput. For each policy, also describe a workload that would benefit from the specific policy.

   •Write-through immediately (no delayed writes)

The highest reliability, since the process doesn't continue until each write is safe on disk. The highest latencies: a process waits each time it writes for the data to be sent to disk. If there's much I/O, processing speed will be tied to the speed of the disk, rather than the speed of the CPU. The worst throughput: all data modified are written to disk, even if later they are overwritten. There's also no time to order the various writes for more efficient disk access. Applications that demand high reliability may want to be sure their data are written to disk. Transactional workloads would be an example.

   •Write-through on close (write modified data to disk when the file is closed. The close operation doesn't return until the write finishes. If a process with open files exits, the system calls the close operation on these files before the exit completes.)

Reliability depends on how long the files are open and how many writes are done to them before they are closed. If files are open for only a short time, reliability may be pretty good. Latencies also depend on how long files are open. If files are open only a short time, then this is almost as bad as "write-through immediately." Disk throughput also depends on how long files are open and often previously-written data is overwritten. If, for example, some section of data is rewritten 10 times before the file is closed, only the last write needs to be

performed. This leaves the disk available for other operations. Workloads with lots of rewriting and files that are open for a long time might benefit. (Unfortunately, even though this policy is used in some well-known systems, UNIX-like workloads often tend to leave files open for only a short time, so this policy doesn't help performance much in that environment.)

- Delay for 30 seconds (write out modified data from the cache when it's 30 seconds old)

This provides only limited reliability, since data that an application believes are already written could be lost for up to 30 seconds. This improves process latencies, since the process never needs to wait synchronously for the data to reach disk. Throughput is improved since the disk writes can be ordered for efficient disk access. In a typical UNIX workload with lots of temporary files, some files and data will have been deleted before the 30 seconds is over. Any modifications to them no longer need to be written through to disk, which improves disk throughput.

- Delay for 5 minutes (write out modified data from the cache when it's 5 minutes old)

The same as just above, only worse for reliability and better for performance.

5. (5 points) In UNIX, there are separate system calls to start a new process in the image of its running parent (`fork`) and to load into a process's address space fresh code and data from an executable that it should use for execution (`exec`). Why doesn't UNIX use just one system call that both creates a new process and loads the desired image (code and data)?

The UNIX scheme is simpler and more flexible. Many system daemons `fork` a copy of themselves, and the child process shares various resources with its parent and can communicate and cooperate with the parent through these shared resources (e.g. file descriptors). If the call always overwrote the process image and created new resources, then this wouldn't be possible. Instead, `exec` can be called when desired by the child process after it has been forked.

(51)