Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Compilers

Date: Wednesday 28th January 2009
Time: 14:00 – 16:00

Please answer any THREE Questions from the FIVE questions provided

The use of electronic calculators is NOT permitted.
1. a) For each item in column one, choose the best match from column two. Each item in column two should be used only once.

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Abstract Syntax Tree</td>
<td>a. Code optimisation</td>
</tr>
<tr>
<td>2. Activation record</td>
<td>b. DFA minimisation</td>
</tr>
<tr>
<td>3. Best’s algorithm</td>
<td>c. Graphical intermediate representation</td>
</tr>
<tr>
<td>4. Bison</td>
<td>d. Instruction scheduling</td>
</tr>
<tr>
<td>5. Chaitin’s heuristic</td>
<td>e. Memory management</td>
</tr>
<tr>
<td>6. Constant folding</td>
<td>f. Parsing</td>
</tr>
<tr>
<td>7. Hopcroft’s algorithm</td>
<td>g. Parsing</td>
</tr>
<tr>
<td>8. List scheduling</td>
<td>h. Register allocation</td>
</tr>
<tr>
<td>9. LR(1) item</td>
<td>i. Register allocation</td>
</tr>
</tbody>
</table>

(5 marks)

b) Give an example of: (i) a lexical analysis error; (ii) a syntax analysis error; (iii) a semantic analysis error. (3 marks)

c) The slice of a program with respect to a given statement, s, and a set of variables, consists of all statements of the program which might affect the value of those variables at point s. Let us assume that you are given an open-source C compiler for a certain processor and you are asked to use it as a basis to build a compiler, which, instead of generating machine code for this processor, produces slices of an input C program. Identify those components that would need to be modified, could be removed, or need to be added to the C compiler you are given in order to produce the slice. (5 marks)

d) The C code fragments below illustrate an optimisation known as unroll-and-jam.

```c
/*before unroll-and-jam*/
for (i=0; i<n; i++)
  for (j=0; j<n; j++)
    a[i,j]=a[i,j-2];

/*after unroll-and-jam*/
for (i=0; i<n; i+=3)
  for (j=0; j<n; j+=3)
    a[i,j]=a[i,j-2];
```

In its general form, for a two-level perfect loop nest, unroll-and-jam unrolls the outer loop multiple times and then jams all the inner loops together.

i) What is the implicit assumption made in the example above, without which the optimisation would not always be valid? (3 marks)

ii) Discuss the impact that an optimisation, such as unroll-and-jam, might have on register allocation and instruction scheduling. (4 marks)
2. a) Consider the alphabet $V=\{0, 1, \ldots, 9\}$ and the language $L$, which consists of all strings of $V$, which represent all integers that are greater than 798 (for example, the strings 799, 890, 2345, 777777 belong to the language $L$, whereas the strings 1, 42, 711, 798 do not). Provide a regular expression that generates all strings of the language $L$. (4 marks)

b) Draw the DFA and write regular expressions that generate the strings recognised by each of the following transition tables:

<table>
<thead>
<tr>
<th>States</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 (start)</td>
<td>-</td>
<td>S1</td>
</tr>
<tr>
<td>S1 (final)</td>
<td>S2</td>
<td>-</td>
</tr>
<tr>
<td>S2</td>
<td>-</td>
<td>S1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0 (start)</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>S1</td>
<td>-</td>
<td>S3</td>
</tr>
<tr>
<td>S2</td>
<td>S3</td>
<td>-</td>
</tr>
<tr>
<td>S3 (final)</td>
<td>-</td>
<td>S3</td>
</tr>
</tbody>
</table>

(4 marks)

c) Draw a minimum state DFA for the regular expression $(a \mid b)^* a (a \mid b)$. (6 marks)

d) There are two kinds of comments in a C++-like language. The first considers as a comment everything that follows a pair of forward slashes, //, until the end of the line (\n character). The second considers as a comment everything which is between /X and the first occurrence of X/.

i) Write a regular expression that would recognise comments in C++ according to the above. You may assume that your input language consists of the symbols /, X, \n, other (where other is defined as any character that is not /, X, \n). (4 marks)

ii) Draw the DFA corresponding to your regular expression in i). (2 marks)
3. a) Consider the following grammar:

1. $S \rightarrow aS$
2. $S \rightarrow Ab$
3. $A \rightarrow XYZ$
4. $A \rightarrow \varepsilon$
5. $X \rightarrow cS$
6. $X \rightarrow \varepsilon$
7. $Y \rightarrow dS$
8. $Y \rightarrow \varepsilon$
9. $Z \rightarrow eS$

i) Give a leftmost derivation of the string $aebc$ and draw the corresponding parse tree. (3 marks)

ii) Does this grammar have the LL(1) property? Justify your answer. (2 marks)

iii) If we add the production rule $X \rightarrow aS$ does the grammar have the LL(1) property? Justify your answer. (2 marks)

b) Consider the grammar and the Action and Goto tables to drive LR(1) parsing:

<table>
<thead>
<tr>
<th></th>
<th>STA</th>
<th>ACTION</th>
<th>GOTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G→S</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>2</td>
<td>S→x</td>
<td>0</td>
<td>S2</td>
</tr>
<tr>
<td>3</td>
<td>S→Ay</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>A→Bx</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>B→z</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Show, in full detail, the steps that an LR(1) parser would follow to parse the string $zxy$ (NB: for each step your answer should show the contents of the stack, what is the next input and the action that is taken). (4 marks)

(Question 3 continues on the following page)
(Question 3 continues from the previous page)

c) Consider the following grammar:

\[
S \rightarrow . L \\
L \rightarrow L B \mid B \\
B \rightarrow 0 \mid 1
\]

The synthesised attribute \( S.\text{val} \) gives the decimal value of the binary fraction generated by \( S \). For example, for the string .101, \( S.\text{val} = 0.625 \).

i) Specify appropriate semantic rules, using only synthesised attributes, to compute the value of \( S.\text{val} \) in this grammar. (6 marks)

ii) Show how your attributes can compute the value of \( S.\text{val} \) for the string .101, by drawing the parse tree and annotating it with the value of the attributes in each case. (3 marks)
4.  

a) Draw the control flow graph of the following C-like code fragment:

```c
if (a==b) { c=d+e; } else { if (c==e) then d=0; }
d=d*2+1;
e=(c+1)*(d+2)*(e+3);
for (i=d; i<=e; i++) a[i]=0;
a=a+1
```

(3 marks)

b) Convert the following, C-like code fragment into three-address code. State your assumptions.

```c
for (i=0; i<5; i++)
a[[b[i]]] = b[a[i]];
```

(4 marks)

c) i) Provide a generic approach that a C compiler might follow to generate code for a C-like `for` loop of the form:

```
for (expr1; expr2; expr3) {
    loop_body;
}
```

(4 marks)

ii) A certain C compiler has generated executable code for the following program. When running the executable, the program falls into an infinite loop when x and y are given the values 2147483644 and 2147483647, respectively, but it works as expected if x and y are given smaller values. Why is this happening? Would this happen if this compiler used your method in i) above to generate code for `for` loops? Justify your answer.

```c
main() {
    int i, x, y;
    scanf("%d", &x);
    scanf("%d", &y);
    for(i=x; i<=y; i++)
        printf(".");
}
```

(4 marks)

d) Provide a general expression that a compiler might use to reference element \([I][J]\) of a byte array of the form \([N...M][P...Q]\), where \(1\leq N\leq M\) and \(1\leq P\leq Q\). The array is stored in row-major order and each element is referenced as a memory offset of the address used to store the first element, that is \([N][P]\), of the array. (Note: \(A[N...M][P...Q]\) means that the rows of the array are indexed by \(N, N+1, N+2, ..., M\) and the columns are indexed by \(P, P+1, P+2, ..., Q\). For example, the array definition \(A[9...12][1...3]\) allocates memory space for a total of 12 elements).

(5 marks)
5.  

a)  Consider the following basic block:

1. load r1, @x  
2. load r2, @y  
3. add r3, r1, r2  
4. mult r4, r1, r2  
5. add r5, r3, 1  
6. add r6, r4, r3  
7. sub r7, r6, r4  
8. mult r8, r5, r7

i) Identify the live ranges for all register values.  
(3 marks)

ii) Draw the interference graph and apply a colouring algorithm of your choice to colour the graph using the smallest number of colours.  
(4 marks)

b) Given the following control flow graph and basic blocks, identify the global live ranges for all values. You only need to show the values that are live on entry to each basic block as well as the values that are live on exit from each basic block.

\[  
\begin{align*}  
1. & A=1  
2. & D=3  
3. & A=7  
4. & C=A+7  
5. & C=9  
6. & D=A+C  
7. & A=A+C+D  
\end{align*}  
\]

(4 marks)

c) Explain how list scheduling can be used to generate a schedule for the following basic block on a processor that can issue up to two instructions per cycle and show the schedule. Assume that all instructions have a latency of one cycle, except \texttt{mult} and \texttt{load}, each of which has a latency of 2 cycles.

1. load r1, @x  
2. \texttt{mov} r2, 7  
3. \texttt{shr} r2, r2, 1  
4. \texttt{mov} r3, 12  
5. \texttt{mult} r3, r3, 19  
6. add r4, r2, r3  
7. sub r5, r3, 1  
8. add r6, r3, 1  
9. \texttt{mult} r7, r2, r1

(5 marks)

(Question 5 continues on the following page)
d) A critical aspect of any scheduling algorithm is the mechanism for setting priorities and for breaking ties when several operations with the same priority are ready at the same cycle. Suggest suitable tiebreakers and explain your rationale. Using as an example the schedule generated in c) above, demonstrate how different tiebrakers can affect the schedule produced. (4 marks)