Two hours

**UNIVERSITY OF MANCHESTER**
**SCHOOL OF COMPUTER SCIENCE**

# Compilers

Date: Wednesday 23rd May 2012

Time: 14:00 - 16:00

**Please answer any THREE Questions from the FIVE questions provided**

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

1. a) For each item in column one, choose the best match for column two. Each item in column two should be used only once.

| Column 1 | Column 2 |
|---|---|
| 1. Abstract Syntax Tree | a. Bottom-Up parsing |
| 2. Access Links | b. Code optimisation |
| 3. Activation Records | c. Context-sensitive analysis |
| 4. Attribute Dependence Graph | d. Graphical intermediate representation |
| 5. Data Dependence Graph | e. Java Virtual Machine |
| 6. Dependence Vector | f. List Scheduling |
| 7. Interference Graph | g. Memory Management |
| 8. Shift/Reduce Conflict | h. Non-local Addressability |
| 9. Stack Machine Code | i. Register Allocation |

(5 marks)

b) A FORTRAN program typically consists of separately compiled functions, known as subroutines. In order to allow two subroutines to share variables one can use a statement known as COMMON, whose syntax is as follows:

COMMON / MYCOM / I1(100,100), I2(100,100), I3(100,100)

In this example, the name of the common memory block is MYCOM and this includes the memory space occupied by the three arrays of integers I1, I2, I3. The space allocated for this common block includes 30000 memory locations. Since the array names are not part of the common block, another subroutine could contain the contain the common statement:

COMMON / MYCOM / I4(30000)

This would mean that the first element of array I1, i.e. I1(1,1), of the first subroutine and the first element of array I4, i.e. I4(1), of the second subroutine occupy the same memory position. Obviously, array elements I3(100,100) and I4(30000) also occupy the same position.

i) Assume that two subroutines share memory using a COMMON statement. One subroutine contains the statement

COMMON / MMMCOM / I1(100,100), I2(100,100), I3(100,100)

and a second subroutine the statement shown below

COMMON / MMMCOM / I4(500,60)

where I4 is also an array of integers. Derive a formula, which for each element of array I4 finds the corresponding array element of the first block.

(6 marks)

ii) Suggest a generic approach that the compiler can follow to determine whether two array elements from two different COMMON statements refer to the same memory location.

(5 marks)

iii) Besides sharing variables can you think of another reason for the existence of the COMMON statement?

(4 marks)

2. a) Consider the alphabet V={0, 1, …, 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:

| | a | b |
|---|---|---|
| S0 (start) | - | S1 |
| S1 (final) | S2 | - |
| S2 | - | S1 |

| | a | b |
|---|---|---|
| S0 (start) | S1 | S2 |
| S1 | - | S3 |
| S2 | S3 | - |
| S3 (final) | - | S3 |

(4 marks)

c) Explain whether it is possible to write regular expressions to recognise: (i) any even integer; (ii) any integer which contains as many 2s as 3s (e.g., 8273 is such an integer); (iii) any real number which contains the same number of digits in both the integer part and the decimal part (e.g., 123.456 is such a number, but 789.01 is not); (iv) any binary number which starts with one or more 1s and is followed by one or more 0s (e.g., 11000). Justify your answer. (4 marks)

d) Provide a regular expression that would recognise unsigned floating point constants such as

1.2 0.23E-8

but would not accept

.123 20. 21.E4 3E4

(that is, the decimal point should always be followed and preceded by a digit) (4 marks)

e) Provide a deterministic finite automaton (DFA) for the regular expression above. (4 marks)

[PTO]

3. a) The grammar of a simple programming language is given by:

1. Goal $\rightarrow$ L
2. L $\rightarrow$ S L
3. L $\rightarrow$ endprogram
4. S $\rightarrow$ id = E
5. S $\rightarrow$ if E then S endif
6. S $\rightarrow$ if E then S else S endif
7. E $\rightarrow$ id
8. E $\rightarrow$ constant
9. E $\rightarrow$ E + id

(anything starting with a lower-case letter is a terminal symbol, including + =)

(i) Is this a left-recursive or a right-recursive grammar? Explain why top-down parsing methods cannot handle left-recursive grammars.

(3 marks)

(ii) Derive a leftmost derivation for the program

```
if xzy then if y then y=1 endif else y=2 endif
x=y
endprogram
```

and show the corresponding parse tree. (6 marks)

(ii) Transform this grammar so that it can be used to construct a top-down predictive parser with one symbol of lookahead. (6 marks)

b) Consider the grammar and the Action and Goto tables below:

(1) G $\rightarrow$ S
(2) S $\rightarrow$ x
(3) S $\rightarrow$ Ay
(4) A $\rightarrow$ Bx
(5) B $\rightarrow$ z

| STA | ACTION | | | | GOTO | | |
|---|---|---|---|---|---|---|---|
| | x | y | z | eof | S | A | B |
| 0 | S2 | | S5 | | 1 | 3 | 4 |
| 1 | | | | accept | | | |
| 2 | | | | R2 | | | |
| 3 | | S6 | | | | | |
| 4 | S7 | | | | | | |
| 5 | R5 | | | | | | |
| 6 | | | | R3 | | | |
| 7 | | R4 | | | | | |

Show, in full detail, the steps that an LR(1) parser would follow to parse the string z x y using the above grammar. For each step, your answer should show the contents of the stack, what the next input is and the action that is taken.

(5 marks)

4. a) Draw the call graph of the following C-like code fragment.

```
void A(int x) { if(x>0) A(x-1); else B(x); }
void B(int y) { if(y>–2) C(y); }
void C(int z) { printf(“%d \n”, z); B(z-2); }
main() { A(1); }
```

How many activation records exist in the stack when the execution of this code reaches the `printf` statement for the first time? Your answer should explain what each activation record in the stack corresponds to.

(5 marks)

b) Consider the following C-like code fragment.

```
b=4; c=1; d=3; n=5; scanf(“%d”,&z);
if (d>7) { c+=d+b+z; printf(“total %d \n”,c); }
q=my_function(z*(c-1));
for(i=10; i<=n; i++)  {  q=q+a[i];  }
printf(“final %d \n”,q);
```

What kind of code transformation techniques can a compiler apply to improve the efficiency of this code fragment? In each case, give the resulting version of the code after each transformation. Your answer should show what the most optimised version you can obtain is. State your assumptions.

(5 marks)

c) Provide generic code that could be used to unroll the body of a loop, such as the one below, s times. You should assume that n can be any positive integer and s is an integer, which is greater than or equal to 1 and less than or equal to n.

```
for (i=0<i<n; i++)
   { loop_body; }
```

(6 marks)

d) Discuss two compiler optimisations, whose impact on program performance would be significantly higher when compiling Java than when compiling C.

(4 marks)

[PTO]

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) Explain why a processor with 4 registers would be sufficient to execute this block without the need to do any spilling. (4 marks)

(ii) Explain how Best's algorithm can be used for register allocation in the above basic block. Show what the basic block would look like after applying the algorithm. (4 marks)

b) 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 `mult` and `load`, each of which has a latency of 2 cycles.

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

(5 marks)

c) Suggest how list scheduling can be used to schedule multiple basic blocks, without any dependence between them, at the same time (assume the existence of multiple functional units).

(4 marks)

d) Suggest an approach to determine the maximum number of functional units that can be used to issue different instructions at the same time for any given basic block (without building a schedule).

(3 marks)

**END OF EXAMINATION**