COMP 36512

Two hours - online

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Compilers

Date: Monday 3rd June 2019

Time: 09:45 - 11:45

This is an online examination. Please answer all TWELVE Questions

© The University of Manchester, 2019

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

1. 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. Dead-code elimination	a. Code generation
2. Interference Graph	b. Code optimisation
3. LL(1) property	c. Context-sensitive analysis
4. LR(1) item	d. Intermediate representation
5. S-attribute grammar	e. Lexical analysis
6. Sethi-Ullman labelling scheme	f. Top-down parsing
7. Stack frame	g. Bottom-up parsing
8. Static Single Assignment (SSA)	h. Register allocation
9. Thomson's construction	i. Storage organisation

(9 marks)

2. Assume that you have developed a basic compiler for a subset of the C language that generates code for Pentium single core processors. Identify the components of the compiler that would need to be modified or enhanced to provide the following capabilities:

   (i) allow C++ (or Java) like comments, that is //

   (ii) unroll loops as much as possible

   Justify your answers. (4 marks)

3. A certain C compiler performs the following optimisation:

```
/* before optimisation */
for(i=2; i<=n-1; i++)
 for(j=2; j<=m-1; j++)
   a[i][j]=(a[i-1][j]+a[i][j-1]+a[i+1][j]+a[i][j+1])/4;


/* after optimisation */
for(j=4; j<=m+n-2; j++)
 for(i=max(2,j-m+1); i<=min(n-1,j-2); i++)
a[i][j-i]=(a[i-1][j-i]+a[i][j-1-i]+a[i+1][j-i]+a[i][j+1-i])/4;
```

   Why is this an optimisation? What is it trying to achieve? Justify your opinion.
   (5 marks)

4. Describe the language denoted by the following regular expression.

   ((a | ε) b*)* a (2 marks)

5. An integer is divisible by 25 if its last two digits are divisible by 25. For example, 75425 is divisible by 25, while 77865 (and, in fact, any integer ending in 65) is not. Write a regular expression to recognise all integers between 39980 and 99999 which are divisible by 25.

(3 marks)

6. Consider the grammar and the Action and Goto tables below.

1. G → L
2. L → L P
3. L → P
4. P → (P)
5. P → ( )

STATE	ACTION			GOTO	
	(	)	eof	L	P
0	S3			1	2
1	S3		accept		4
2	R3		R3		
3	S6	S7			5
4	R2		R2		
5		S8			
6	S6	S10			9
7	R5		R5		
8	R4		R4		
9		S11			
10		R5			
11		R4			

(i) Show, in full detail, the steps that an LR(1) parser would follow to parse the string (( )) ( ) 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. (4 marks)

(ii) Consider extending the LR(1) case to the more general cases of using 2 or 3 lookahead symbols. In these cases, for the grammar above, how many columns would the corresponding action/goto table have? Justify your answer. (4 marks)

7. A simple language allows program/variable names that consist of one, two, or three characters, with each character being any of the 26 letters of the latin alphabet (this makes it a total of $26+26^2+26^3 = 18278$ possible names). A simple compiler for this language implements a 1024-entry long symbol table. Suggest a good hash function to map program names onto this symbol table. State any assumptions you make.

(5 marks)

[PTO]

8. Consider 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. (3 marks)

9. 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);
```

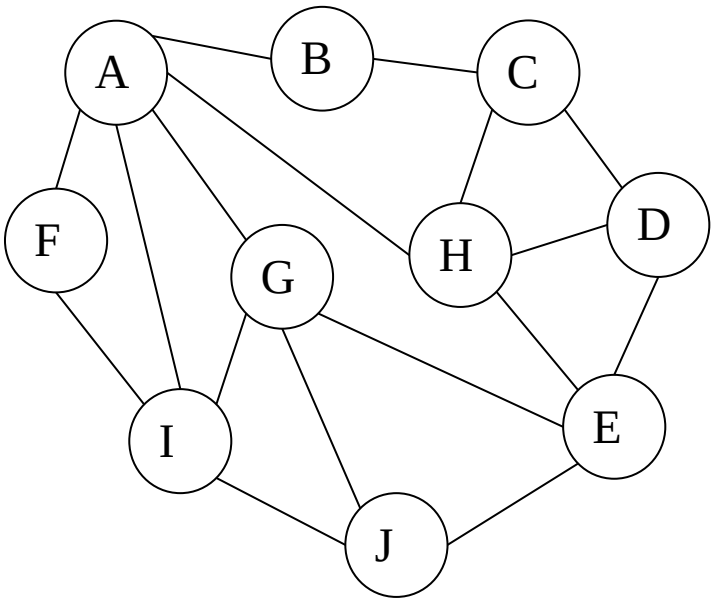
What kind of code transformation techniques can a compiler apply to improve the efficiency of this code fragment? In each case, give the resulting version of the code after each transformation. Your answer should show what the most optimised version you can obtain is. State your assumptions. (6 marks)

10. Identify live ranges for all register values of the following basic block.

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

(3 marks)

11. Apply a graph colouring algorithm of your choice to colour the following interference graph. Explain the steps you take. What is the least number of colours needed?



(4 marks)

12. Explain how list scheduling can be used to generate a schedule for the following basic block and show the schedule. First, assume a processor that can issue up to three instructions per cycle, where at most one instruction is a `load`, at most one instruction is a `mult` and at most one instruction is an `add` or a `mov`. All instructions have a latency of one cycle.

```
1. load r1, @x
2. load r2, @y
3. load r3, @z
4. mov  r4, 7
5. add  r5, r4, r1
6. mult r7, r1, r4
7. mult r8, r2, r5
8. add  r9, r8, r3
```

Then, show the schedule for another processor that can issue up to two instructions per cycle (any instructions). Compare the two schedules you derived.

(8 marks)

END OF EXAMINATION