INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

RH

Date .....FN / AN *Time:* ~~2~~3 Hrs. *Full Marks* 75 *No. of Students* 02
Supplementary Examination, 2010-2011 Deptt. CSE Sub No. CS31003
3rd Yr. B. Tech.(Hons.) / ~~B. Arch.~~ / ~~M. Sc.~~ Sub. Name COMPILERS

Instructions : **Answer all questions**

1. **[10 + 5 + 5]**

   (a) Compute the collection of the set of $LR(0)$ items and construct the SLR parsing table for the following grammar with E as the start symbol.

$$\begin{aligned} E &\rightarrow T + E \mid T \\ T &\rightarrow F * T \mid F \\ F &\rightarrow id \mid *id \mid +id \mid (E) \end{aligned}$$

   (b) What warning will be generated by `bison` if '$F \rightarrow * \; id$' and '$F \rightarrow + \; id$' in the grammar of (**1a**) are replaced by '$F \rightarrow id \; *$' and '$F \rightarrow id \; +$' respectively? Can the warning be stopped by specifying some 'precedence' to the *operators*?

   (c) Give an *extended regular expression* for the declaration of one variable. The type may be either `int` or `float`. The object may be a scalar or a multi-dimensional array.

2. Consider the following program. **[5 + 10 + 10 + 10]**

```
// This program is for Q2
code
   var int r1 r2 r3 a[3][5][4] b[3][5][4] i j k ;
//
// Code for data read is removed
//
i <- 0
   while i < r1: start
       j <- 1
       while j < r2: start
           k <- 2
           while k < r3: start
               b[i][j][k] <- b[i][j][k]+2*a[i][j][k]
               k <- k + 1
           .
           j <- j + 1
       .
       i <- i + 2
   .
end
```

   (a) Answer following questions in connection to the variable declaration of the above program.

      i. What will be the size of the stack frame (variables are allocated space on the run-time stack) for this program?
      Clearly mention the offsets of different objects with respect to the stack pointer (`esp` of `i686`)

      ii. Write `x86` assembly code (GCC on Linux) to saves the base pointers of the old stack frame and to create the new stack frame for the program.

   (b) Translate the high level code to a sequence of *3 address codes*

(c) Consider the largest basic block, construct its DAG and improve the *3-address code* using standard basic block optimization techniques e.g. *elimination of common subexpression, copy propagation, dead-code elimination, strength reduction* etc. Give a brief explanation in each case.

(d) Translate the improved *3-address code* sequence to an assembly language program of *i686, GCC* (as faithfully possible). Use the offsets of objects you specified in the first part.

Registers	*Usage*
eax	*4-byte return value, caller saved*
ecx, edx	*caller saved*
ebp, edx, esi, edi	*callee saved*

A few x86 *Instructions*
`addl, andl, call, cmpl, jmp, je, jg, jl, jne, leal, movl, ret, popl, pushl, sall` (shift left arithmetic), `subl`

3. Write short notes on the following topics. [4 × 5]

(a) Construction of DFA from *regular expression.*

(b) Ambiguous operator grammar for arithmetic expression and LALR parsing.

(c) Register allocation and assignment.

(d) Data-flow analysis.

Sig.of the Paper-Setter ....Goutam Biswas.....