INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

Date: __-___-12 FN/AN	**Time: 3 Hrs**	**Full Marks: 100**	**Deptt. Computer Sc. & Engg.**
No. of Students: 111		**End Autumn Semester Examination, 2012-13**	
Subject No: CS31003		**Subject Name: Compiler Design**	
3rd Year B.Tech. (H)		**Instruction: Attempt *all* questions**	

---

1. Consider `switch` statement of `C` language with the syntax as follows:

```
switch (E) {
      case v1: S1
      case v2: S2
      ...
      case vn-1: Sn-1
      default: Sn
}
```

where `E` is an integer (`int`) expression, $v_1, v_2, \ldots, v_{n-1}$ are (distinct) integer constants and $S_1, S_2, \ldots, S_{n-1}$ are list of statements. The semantics of `switch` statement is defined as in `C`.

a. Define a suitable grammar for `switch` statement. Assume v_i's as literals. There can be zero or more `case` clauses and zero or one `default` clause. For simplicity you may assume that all `case` clauses precede the `default` clause. **[10+**

Note: You do not need to write the production rules for `E` and `S`.

b. Propose a translation scheme for `switch` statement based on your grammar. Show the semantic actions for every production of the grammar. **10+**

Note: There is no need to show the actions for `E` and `S` as you have not expanded the production rules for them.

c. Generate the 3-address codes for the following sample: **10=**

```
switch (x) {
      case 1:  y = x; break;
      case -1: y = -x; break;
      default: y = 1; break;
}
```

30]

Note: Plug in the codes for assignment and break statements from your general understanding of 3-address translation as there are no explicit production rules in your grammar generating these statements. Also assume that variables `x` and `y` have been declared earlier and hence have been added to an appropriate symbol table. Make further assumptions as needed and document them clearly.

2. Consider the following grammar G:

F → int id(P) { L }
P → P, D | D
D → int id
L →L S | S
S → D; | id = id; | if (B) S else S | return id;
B → id relop id

a. Write the semantic actions for the above grammar rules for syntax directed translation to 3-address codes. **[25+**

Note:
- Assume semantics of C language.
- Modify / Augment the grammar as you need.
- Define the attributes of all terminals and non-terminals that you use in writing the semantic actions.
- Define the data structures and functions you use in writing the semantic actions. There is no need to provide their codes for implementation. Just brief English descriptions would do.

b. Propose a scheme to translate your 3-address codes to x86 target. The scheme should clearly highlight the following: **15+**
- Mapping of 3-address statements to assembly instructions.
- Register allocation and Address binding for the parameters and local variables of the function.
- Handling of labels.

c. Consider the following function:

```
int middle(int a, int b, int c)
{
    int r;
    if (a > b)
            if (b > c) r = b;
            else if (a > c) r = c; else r = a;
    else
            if (a > c) r = a;
            else if (b > c) r = c; else r = b;

    return r;
}
```

i. Construct the parse tree for the above code using the grammar. You may show the parse tree in two or three parts if it gets large. **(10+20)=**

ii. Using your schemes in (a) and (b) translate the above code to x86 assembly. **70]**