Department of Computer Science and Engineering
IIT Roorkee
End-Term Examination – CSN 352 Compiler Design

Date: 13-May 2025 | Duration: 3 Hours | Max Marks: 100

Instructions

- Irrelevant content may result in a penalty of upto 25% of the marks for that question
- If you do not know the answer, you may write "I DON'T KNOW" to receive 10% of the marks for that question.

Q.1. (15 points) The use of pragmas is a well-established technique in C/C++ compilers. For example, in Listing 1, modern compiler will generate code for the loop based on openmp target.

```c
int main() {
    #pragma omp parallel for
    for (int i = 0; i < 10; i++) {
        printf("Thread %d processes iteration %d\n", omp_get_thread_num(), i);
    }
    return 0;
}
```

Listing 1: Parallel For Loop using OpenMP

Suppose you are working in an industry where power efficiency is the highest priority. As a compiler expert, you are tasked with addressing the following problem: "If certain parts of the code do not require precise results from floating-point operations, then generate the code assuming those variables are integers; otherwise, generate the accurate floating-point code."

Solve the above problem using pragma by explaining each step of Analysis-Synthesis model.

Q.2. (15 points) We propose a new analysis called "Saint Variable Analysis". A variable is termed a saint variable if it is computed solely from constants or other saint variables, and it is not used in the program. Design a data-flow analysis (DFA) to identify such variables for optimization purposes. Additionally, a variable is classified as a super saint if it is neither used in program nor depends on any other variable (even on saint variables). Provide a separate DFA formulation to detect these variables.

Q.3. (10 points) Define register pressure, and name three compiler optimizations that help in reducing it? Provide an example for each.

Q.4. (10 points) A student mentioned, "Understanding the layout of an activation record may allows you to bypass many concepts (like dangling pointers)". What is an activation record, and how does this statement apply? Explain using a C program example.

Q.5. (10 points) Convert the Listing 2 in Static Single Assignment form.

```c
void main() {
    int i;
    scanf("%d",&i);
    if(i == 0) ans = 100;
    else ans = 200;
    printf("%d\n",ans);
}
```

Listing 2: C Program

Q.6. (10 points) Calculate the minimum number of registers required to execute the Listing 3 on a stack-based machine. Provide a clear justification for your answer.

```c
int binary_search(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key) return mid;
        else if (arr[mid] < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}
```

Listing 3: Binary Search in C

Q.7. (20 points) After seeing your **GitHub** repo, compiler industry became fan of your expertise in optimization technique. Explain each of the optimizations, **step-by-step** that was used to convert the Listing 4 into the Listing 5.

```c
int getValue() {
    return 5;
}
void main() {
    int sum = 0;
    for (int i = 0; i < 2; i++) {
        sum += 1;
    }
    int count = 0;
    while (count < 3) {
        sum += 1;
        count++;
    }
    sum += getValue();
    printf("%d\n", sum);
}
```

Listing 4: C Program with for and while loops

```
function: main
t1 = 10
param t1
call printf
```

Listing 5: Optimized Three-Address Code

Q.8. (10 points) Why is it important to study compilers? What might a computer engineer miss without this knowledge? What have you personally learned from the course?