Assignment No 2.

Q.1 Translate following statement into Three address code if $x < y$ and $a > b$

Ans:-
1. $t_1 = x < y$
2. $t_2 = a > b$
3. $t_3 = t_1 \$\$ t_2$
4. if t_3 go to 5
5. .....

Explanation :-

* $t_1 = x < y$ compares the values of x and y and assigns the result to a new temporary variable t_1
* $t_2 = a > b$ compares the values of a and b and assigns the result to new temporary variable etc.
* $t_3 = t_1 \$\$ t_2$ checks if both t_1 and t_2 are true (ie have a value of 1) and assigns the result to a new temporary variable t_3. If both are true t_3 will be assigned the value 1, otherwise 0.
* If t_3 go t_5 checks if t_3 is true (ie 1 has a value of 1). If is true, The control jumps to the instruction labeled 5. otherwise, the program continues to the next instruction after this block.

Q. 2 Differentiate between loop optimization and local optimization.

Ans:-

Loop optimization	Local optimization
1. Optimizes loop in the code.	1. Optimizes code within a single function.
2. Reduce the number of instructions executed in loops.	2. Reduce the number of instruction executed in a function.
3. Maximizes the use of processor registers in loops.	3. Eliminates redundent code with a function.
4. Improves performance of loop-intensive code	4. Improve performance of code by minimizing memory accesses.
5. Techniques used include loop unrolling loop fusion loop-invarient code motion and loop vectorization	5. Techniques used include constant folding dead code elimination strength reduction and common subexpression elimination

Q.3 Explain Peephol optimization with their characterstics.

Ans:- Peephole optimization is local optimization technique used in compilers to improve code performance. It focuses on eliminating redundent or inefficient code sequence in a small and fixed -size window of instructions, known as a "peephole"

Characterstics of peephole optimization :-

1. localized optimization :- Peephole optimization is a localized optimization technique that focusses on optimizing a small and fixed - size window of instructions. This window typically between three and ~~seven~~ seven instruction large.

2. Iterative process :- Peephole optimization is an iterative process that examines each instruction in the window and looks for optimization to eliminate redundent.

3. Machine Independent :- Peephole optimization is a machine-independent optimization technique which means that it can be applied to code generated for any target machine architecture.

Q.4. What are issues in generating target code?

Ans:- Generating target code is a complex process that involves translating the source code into machine code that can be executed by the target machine.

1. Target machine architecture :- The target architecture may not support all the feature and construct of the source language. This can result in a loss of functionality or performance degradation in the generated code.

2. Instruction set limitations :- The target machine instruction set may have limitation that make it difficult to generate efficient code. For example - Some machine may not support certain types of memory access or have limited supported for floating-point arithmetic.

3. Register allocation :- Register allocation is a critical issues in generating efficient code.

If the compiler does not allocate registers optimally, it can result in inefficient code that requires excessive memory access.

4. Code size :- The size of the generated code is an important consideration, especially for embedded system or mobile device with limited memory. If the generated code is too large it may not fit in the available memory or require excessive memory access, leading to performance degradation.

5. Optimization tradeoffs :- The compiler must make trade offs between generating efficient code and generating code quality. This can result in suboptimal code in some cases.

6. Debugging :- Debugging generated code can be challenging, especially if the generated code does not match the source code exactly.

Q.5 Construct the given expression into DAG.

a + a * (b - c) + (b - c) * d

Ans:- Steps for constructing a DAG :-

1. d_1 = leaF(id, entry = a)
2. d_2 = leaF(id, entry - a) = d_1
3. d_3 = leaF(id, entry - b)
4. d_4 = leaF(id, entry - c)
5. d_5 = node('-', d_3, d_4)
6. d_6 = node('*', d_1, d_5)
7. d_7 = node('+', d_1, d_6)
8. d_8 = leaF(id, entry - b) = d_3
9. d_9 = leaF(id, entry - c) = d_4
10. d_{10} = node('-', d_3, d_4) = d_5
11. d_{11} = leaF(id, entry - d)
12. d_{12} = node('*', d_5, d_{11})
13. d_{13} = node('+', d_7, d_{12})

Now

$t_1 = b - c$

$t_2 = a * t_1$

$t_3 = t_1 * d$

$t_4 = a + t_2$

$t_5 = t_4 + t_3$

DAG :-