Problem 1

- Show the renamed version of the following code. Assume that you have 4 rename registers T1-T4.

```
R1 ← R2+R3
R3 ← R4+R5
BEQZ R1
R1 ← R1 + R3
R1 ← R1 + R3
R3 ← R1 + R3

T1 ← R2+R3
T2 ← R4+R5
BEQZ T1
T4 ← T1+T2
T1 ← T4+T2
T2 ← T1 +R3
```
Problem 2

- Show the renamed version of the following code. Assume that you have 36 physical registers and 32 architected registers.

\[
\begin{align*}
R1 & \leftarrow R2 + R3 \\
R3 & \leftarrow R4 + R5 \\
\text{BEQZ } & R1 \\
R1 & \leftarrow R1 + R3 \\
R1 & \leftarrow R1 + R3 \\
R3 & \leftarrow R1 + R3 \\
R4 & \leftarrow R3 + R1 \\
P33 & \leftarrow P2 + P3 \\
P34 & \leftarrow P4 + P5 \\
\text{BEQZ } & P33 \\
P35 & \leftarrow P33 + P34 \\
P36 & \leftarrow P35 + P34 \\
P1 & \leftarrow P36 + P34 \\
P3 & \leftarrow P1 + P36
\end{align*}
\]
The Dataflow Model

- Von Neumann model: An instruction is fetched and executed in **control flow order**
  - As specified by the **instruction pointer**
  - Sequential unless explicit control flow instruction

- Dataflow model: An instruction is fetched and executed in **data flow order**
  - i.e., when its operands are ready
  - i.e., there is **no instruction pointer**
  - Instruction ordering specified by data flow dependence
    - Each instruction specifies “who” should receive the result
    - An instruction can “fire” whenever all operands are received
  - Potentially many instructions can execute at the same time
    - Inherently more parallel
von Neumann vs Dataflow

• Consider a von Neumann program
  – What is the significance of the program order?
  – What is the significance of the storage locations?

\[ v \leftarrow a + b; \]
\[ w \leftarrow b * 2; \]
\[ x \leftarrow v - w \]
\[ y \leftarrow v + w \]
\[ z \leftarrow x * y \]

Sequential

• Which model is more natural to you as a programmer?
More on Data Flow

• In a data flow machine, a program consists of data flow nodes
  – A data flow node fires (fetched and executed) when all it inputs are ready
    • i.e. when all inputs have tokens

• Data flow node and its ISA representation
Dataflow Nodes

- A small set of dataflow operators can be used to define a general programming language
Dataflow Graphs

{x = a + b;
  y = b * 7
in
  (x-y) * (x+y)}

- Values in dataflow graphs are represented as tokens
  token < ip, p, v >
  instruction ptr port data

- An operator executes when all its input tokens are present; copies of the result token are distributed to the destination operators

no separate control flow
Control Flow vs. Data Flow

\[ a := x + y \]
\[ b := a \times a \]
\[ c := 4 - a \]

**Figure 2.** A comparison of control flow and dataflow programs. On the left a control flow program for a computer with memory-to-memory instructions. The arcs point to the locations of data that are to be used or created. Control flow arcs are indicated with dashed arrows; usually most of them are implicit. In the equivalent dataflow program on the right only one memory is involved. Each instruction contains pointers to all instructions that consume its results.
Data Flow Characteristics

• Data-driven execution of instruction-level graphical code
  – Nodes are operators
  – Arcs are data (I/O)
  – As opposed to control-driven execution

• Only real dependencies constrain processing

• No sequential instruction stream
  – No program counter

• Execution triggered by the presence/readiness of data
A Dataflow Processor

- **Token** = **Data1** + **Tag** + **Destination**

- **Matching Area**

- **Group** = **Data1** + **Data2** + **Tag** + **Destination**

- **Instruction Fetch Area**

- **Execution Package** = **Data1** + **Data2** + **OpCode** + **Tag** + **Destination**

- **Data Flow Proc. Element**

- **Token** = **Data** + **Tag** + **Destination**

- **Pool of Unmatched Tokens**

- **OP** → **Dest.**

- **New One**
Data Flow Advantages/Disadvantages

• Advantages
  – Very good at exploiting irregular parallelism
  – Only real dependencies constrain processing

• Disadvantages
  – No precise state
    • Debugging very difficult
    • Interrupt/exception handling is difficult
  – Bookkeeping overhead (tag matching)
  – Too much parallelism? (Parallelism control needed)