Basic Blocks
what's a basic block?

- A basic block is a straight-line piece of code with no control flow
- Basic rule: once you execute the first instruction of the basic block, you are guaranteed to execute all the instructions of the basic block
  - No way to exit out of the basic block before the end (no jump statements)
  - No way to enter the basic block after the beginning (no labels you can jump to)
- Control transfers occur between basic blocks

```
ADD t7, t1, t2
Lab1:
ADD t9, t1, t3
SUB t2, t7, t9
BNE t2, t1 Lab1
ADD t2, t4, t7
```
why a basic block?

• When we are optimizing code a key question we want to answer is: *will the transformed code behave the same as the original code?*
  • Must be true no matter how the program executes, no matter what input the program sees
  • If I can’t guarantee this, I can’t do the transformation!
• It is much easier to reason about the behavior of straight-line code than it is to reason about code with jumps and branches

```
ADD t7, t1, t2
Lab1:
ADD t9, t1, t3
SUB t2, t7, t9
BNE t2, t1 Lab1
ADD t2, t4, t7
```
more formally

- A basic block is a maximal sequence of instructions $I_0, I_1, I_2, \ldots, I_n$ such that if $I_j$ and $I_{j+1}$ are two adjacent statements in this sequence, then
  - The execution of $I_j$ is always immediately followed by the execution of $I_{j+1}$
  - The execution of $I_{j+1}$ is always immediately preceded by the execution of $I_j$

```
ADD t7, t1, t2
Lab1:
  ADD t9, t1, t3
    SUB t2, t7, t9
  BNE t2, t1 Lab1
ADD t2, t4, t7
```
more formally

- A basic block is a maximal sequence of instructions $I_0, I_1, I_2, \ldots, I_n$ such that if $I_j$ and $I_{j+1}$ are two adjacent statements in this sequence, then
  - The execution of $I_j$ is always immediately followed by the execution of $I_{j+1}$
  - The execution of $I_{j+1}$ is always immediately preceded by the execution of $I_j$

```
ADD t7, t1, t2
Lab1:
  ADD t9, t1, t3
  SUB t2, t7, t9
  BNE t2, t1 Lab1
  ADD t2, t4, t7
```
A basic block is a maximal sequence of instructions $I_0, I_1, I_2, ..., I_n$ such that if $I_j$ and $I_{j+1}$ are two adjacent statements in this sequence, then

- The execution of $I_j$ is always immediately followed by the execution of $I_{j+1}$
- The execution of $I_{j+1}$ is always immediately preceded by the execution of $I_j$
more formally

• A basic block is a maximal sequence of instructions $I_0, I_1, I_2, \ldots, I_n$ such that if $I_j$ and $I_{j+1}$ are two adjacent statements in this sequence, then
  • The execution of $I_j$ is always immediately followed by the execution of $I_{j+1}$
  • The execution of $I_{j+1}$ is always immediately preceded by the execution of $I_j$

|ADD t7, t1, t2 |
|Lab1: |
|ADD t9, t1, t3 |
|SUB t2, t7, t9 |
|BNE t2, t1 Lab1 |
|ADD t2, t4, t7 |
more formally

• A basic block is a maximal sequence of instructions $I_0, I_1, I_2, \ldots, I_n$ such that if $I_j$ and $I_{j+1}$ are two adjacent statements in this sequence, then
  • The execution of $I_j$ is always immediately followed by the execution of $I_{j+1}$
  • The execution of $I_{j+1}$ is always immediately preceded by the execution of $I_j$

```
ADD t7, t1, t2
Lab1:
  ADD t9, t1, t3
  SUB t2, t7, t9
  BNE t2, t1 Lab1
ADD t2, t4, t7
```
finding basic blocks

- Use three-address code
- Jump targets are labeled
- Also label beginning/end of functions
- Want to keep track of targets of jump statements
  - Any statement whose execution may immediately follow execution of jump statement
  - **Explicit target:** targets mentioned in jump statement
  - **Implicit target:** statements that follow conditional jump statements
    - The statement that gets executed if the branch is not taken
finding basic blocks

\[
\begin{align*}
A &= 4 \\
t1 &= A \times B \\
do \{ \\
& \quad t2 = t1/C \\
& \quad \text{if } (t2 \geq W) \{ \\
& \quad \quad M = t1 \times k \\
& \quad \quad t3 = M + I \\
& \quad \} \\
& \quad H = I \\
& \quad M = t3 - H \\
\} \text{ while } (T3 \geq 0)
\end{align*}
\]
finding basic blocks

1. \( A = 4 \)
2. \( t1 = A \times B \)
3. **L1:** \( t2 = t1 / C \)
4. if \( t2 < W \) goto L2
5. \( M = t1 \times k \)
6. \( t3 = M + I \)
7. **L2:** \( H = I \)
8. \( M = t3 - H \)
9. if \( t3 \geq 0 \) goto L3
10. goto L1
11. **L3:** halt
finding basic blocks

• Step 1: Identify **leaders**: first statement of a basic block
• Step 2: In program order, construct a block by appending subsequent statements up to, but not including, the next leader

• Identifying leaders
  • First statement in the program
  • Explicit target of any conditional or unconditional branch
  • Implicit target of any branch
partitioning algorithm

- Input: set of statements, \( stat(i) = \text{ith statement in input} \)
- Output: set of leaders, set of basic blocks where \( block(x) \) is the set of statements in the block with leader \( x \)
- Algorithm

\[ leaders = \{1\} \quad //\text{Leaders always includes first statement} \]

\[ \text{for } i = 1 \text{ to } |n| \quad //|n| = \text{number of statements} \]

\[ \text{if } stat(i) \text{ is a branch, then} \]

\[ leaders = leaders \cup \text{all potential targets} \]

\[ \text{end for} \]

worklist = leaders

\[ \text{while worklist not empty do} \]

\[ x = \text{remove earliest statement in worklist} \]

\[ block(x) = \{x\} \]

\[ \text{for } (i = x + 1; i \leq |n| \text{ and } i \not\in leaders; i++) \]

\[ block(x) = block(x) \cup \{i\} \]

\[ \text{end for} \]

\[ \text{end while} \]
where are the basic blocks?

1. \[ A = 4 \]
2. \[ t1 = A \times B \]
3. \[ \textbf{L1:} \quad t2 = t1 / C \]
4. \[ \text{if } t2 < W \text{ goto L2} \]
5. \[ M = t1 \times k \]
6. \[ t3 = M + I \]
7. \[ \textbf{L2:} \quad H = I \]
8. \[ M = t3 - H \]
9. \[ \text{if } t3 \geq 0 \text{ goto L3} \]
10. \[ \text{goto L1} \]
11. \[ \textbf{L3:} \quad \text{halt} \]
where are the basic blocks?

leader 1 A = 4
leader 2 t1 = A * B
leader 3 L1: t2 = t1 / C
leader 4 if t2 < W goto L2
leader 5 M = t1 * k
leader 6 t3 = M + I
leader 7 L2: H = I
leader 8 M = t3 - H
leader 9 if t3 ≥ 0 goto L3
leader 10 goto L1
leader 11 L3: halt
where are the basic blocks?

| leader | 1 | A = 4 |
| leader | 2 | t1 = A * B |
| leader | 3 | t2 = t1 / C |
| leader | 4 | if t2 < W goto L2 |
| leader | 5 | M = t1 * k |
| leader | 6 | t3 = M + I |
| leader | 7 | H = I |
| leader | 8 | M = t3 - H |
| leader | 9 | if t3 ≥ 0 goto L3 |
| leader | 10 | goto L1 |
| leader | 11 | L3: halt |
next: control flow graphs