Worksheet-11 Solution

(From Lecture given on 02/18/2019)

- Consider the following sequence of IR instructions:

```
1. t1 := i × 4
2. t2 := a + t1
3. t3 := *t2          // Load instruction
4. t4 := b + t1
5. *t4 := t3          // Store instruction
```

- Show the output of instruction selection with the minimal total cost, using any approach you like (need not be an algorithm presented in class). The output should be a sequence of machine instructions. You can generate instructions that refer to virtual IR registers such as i, a, b, and t1…t4.

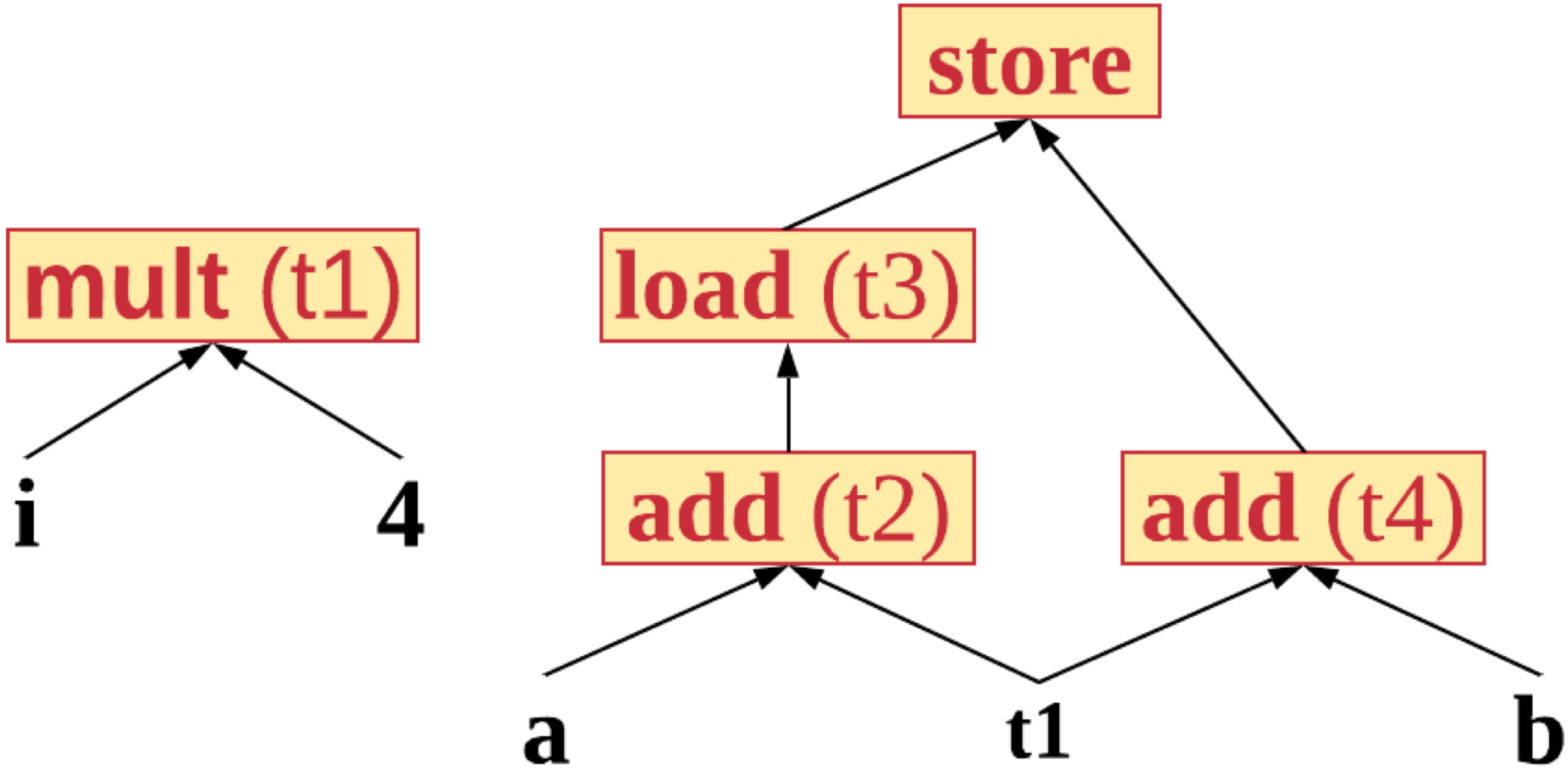
The ISA that we are targeting includes the following instructions, with costs:

- **add r2, r1** r1 ← r1 + r2 Cost = 2
- **addi c, r1** r1 ← r1 + c Cost = 2
- **muli c, r1** r1 ← r1 × c Cost = 4
- **lshi c, r1** r1← r1 << c // Left shift r1's content by c bits.
  // Each shift equals multiplication by 2 Cost = 2
- **lw r2, r1** r1 ← *r2 // Load a word from address contained in r2 to r1 Cost = 6
- **sw r2, r1** *r1 ← r2 // Store a word in r2 into address contained in r1 Cost = 4
- **movmw r2, r1** *r1 ← *r2 // Copy a word from address in r2 to that in r1 Cost = 8

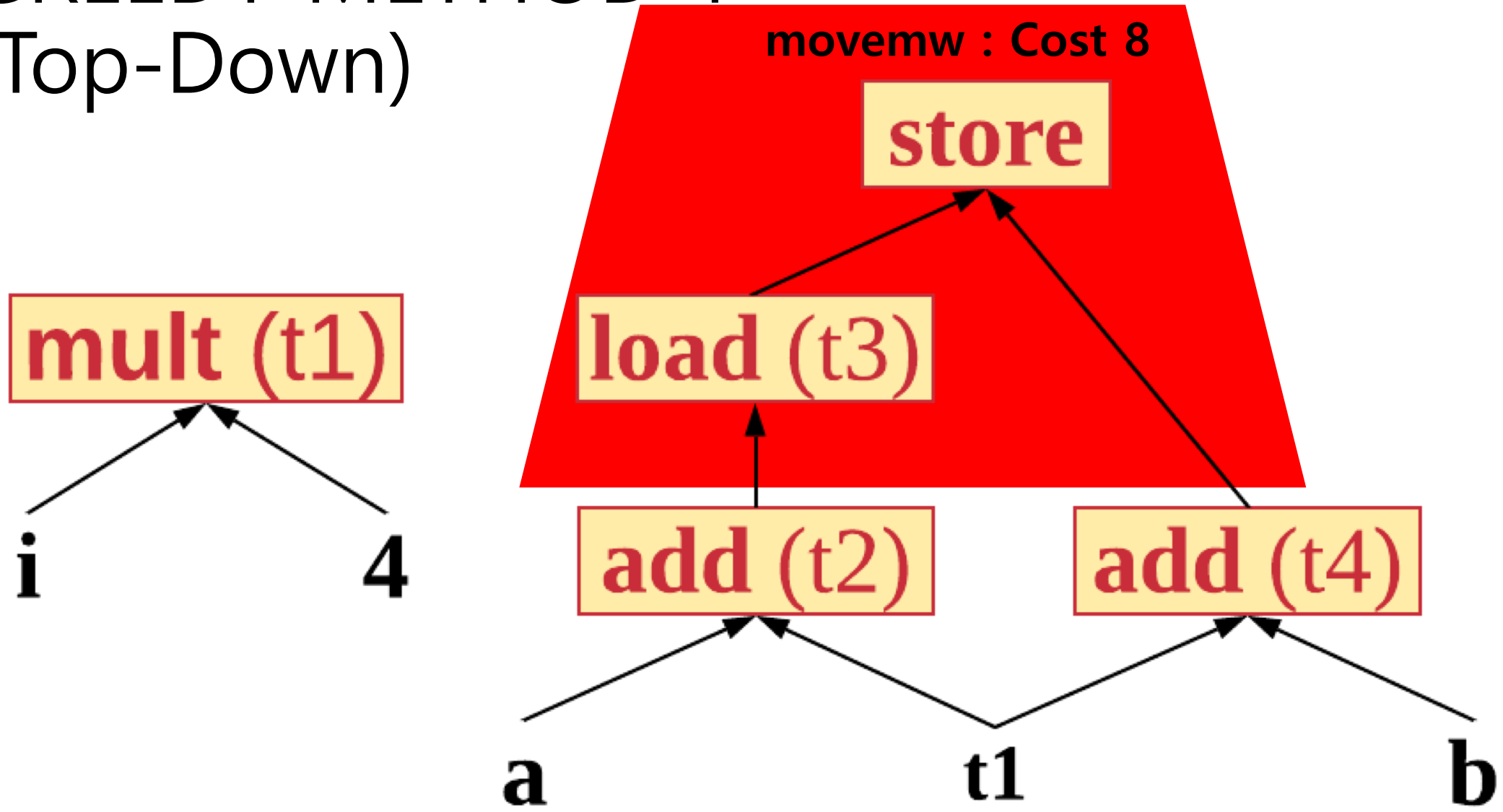
Most naïve solution : Generate machine instructions for each IR instruction

TOTAL COST : 16

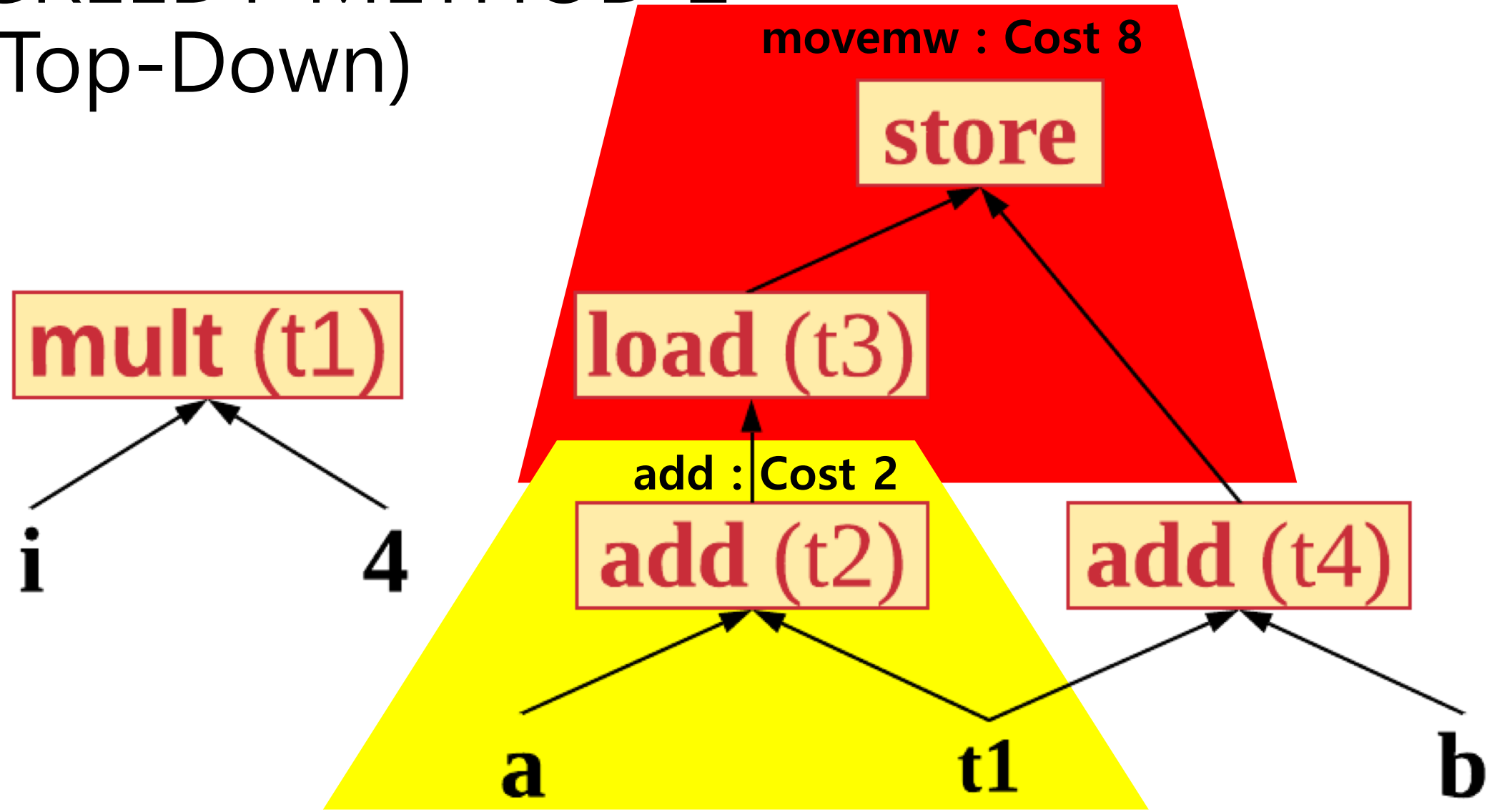
Dependence Tree for the given IR instructions



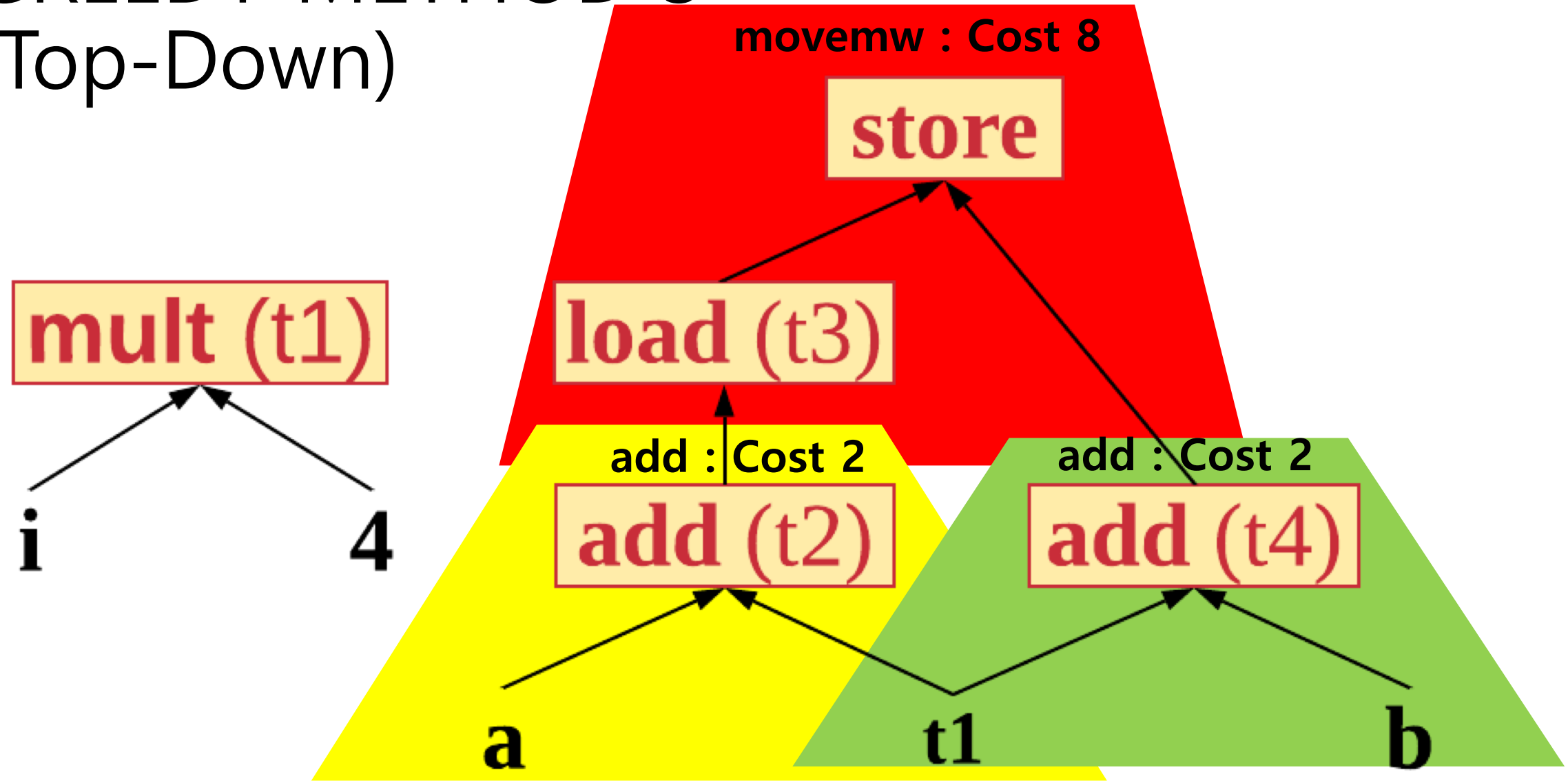
GREEDY METHOD 1 (Top-Down)



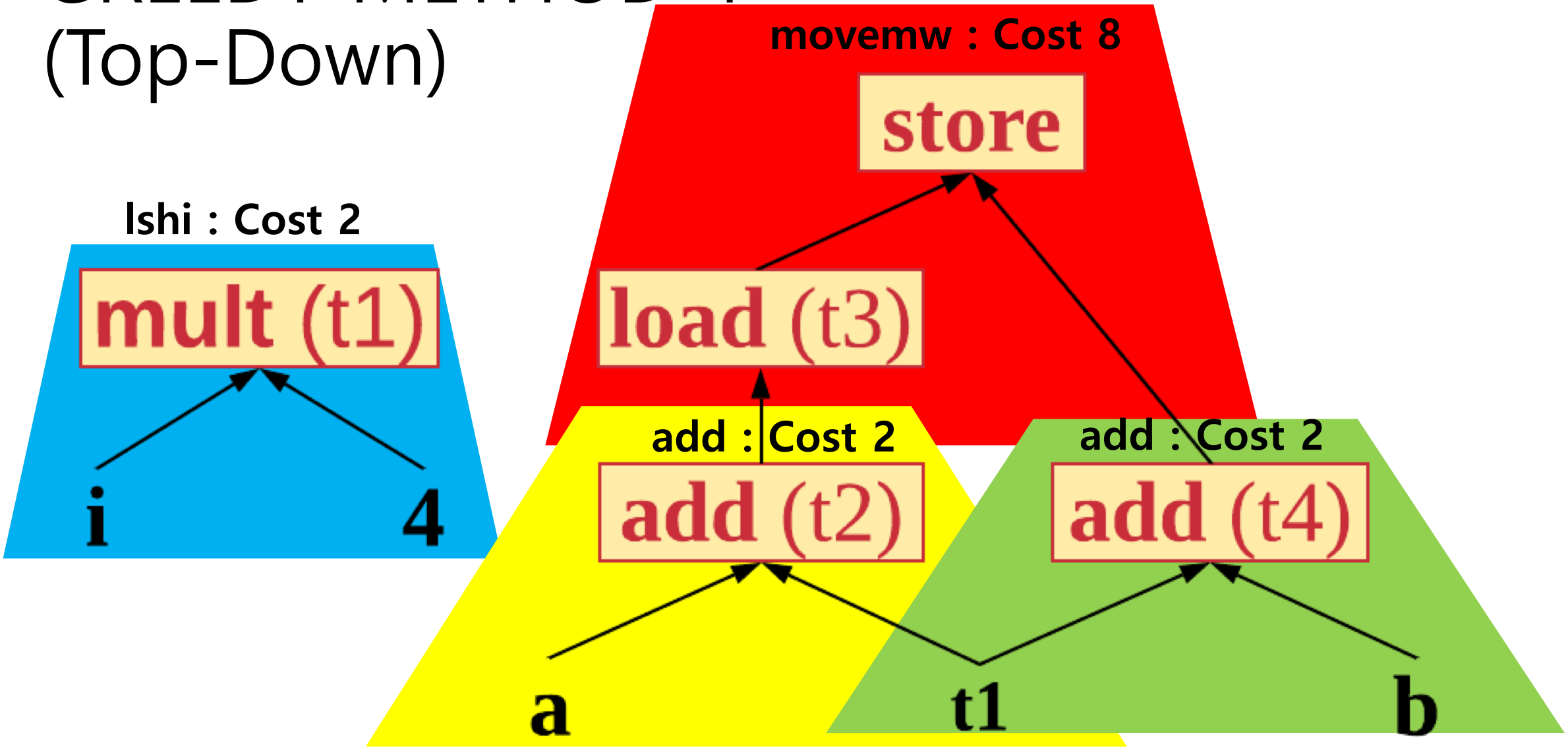
GREEDY METHOD 2 (Top-Down)



GREEDY METHOD 3 (Top-Down)



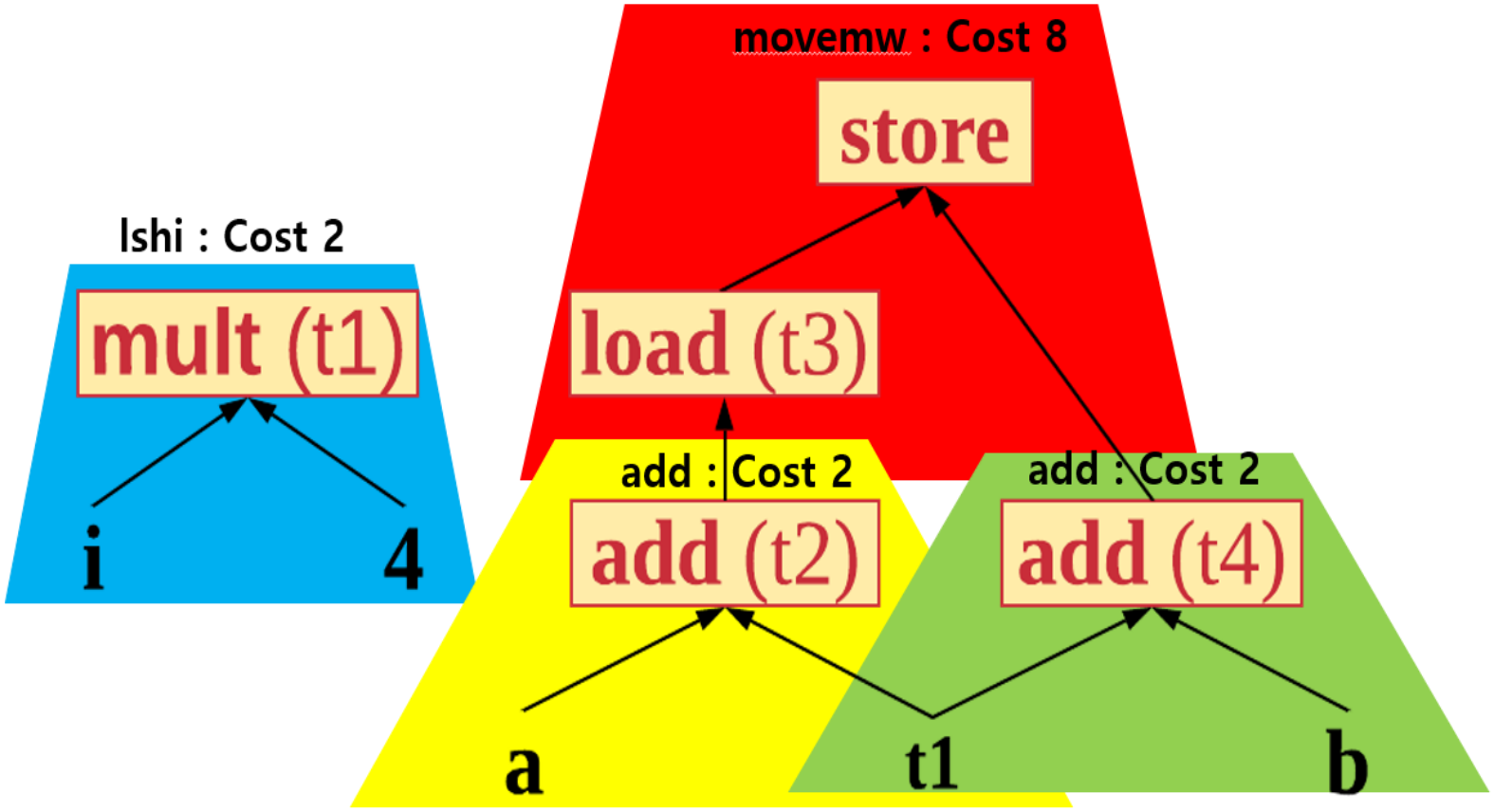
GREEDY METHOD 4 (Top-Down)



GREEDY METHOD 5 (Top-Down)

- All nodes are covered with tiles.

1. lshi 2, i cost 2
2. add i, a cost 2
3. add i, b cost 2
4. movmw a, b cost 8

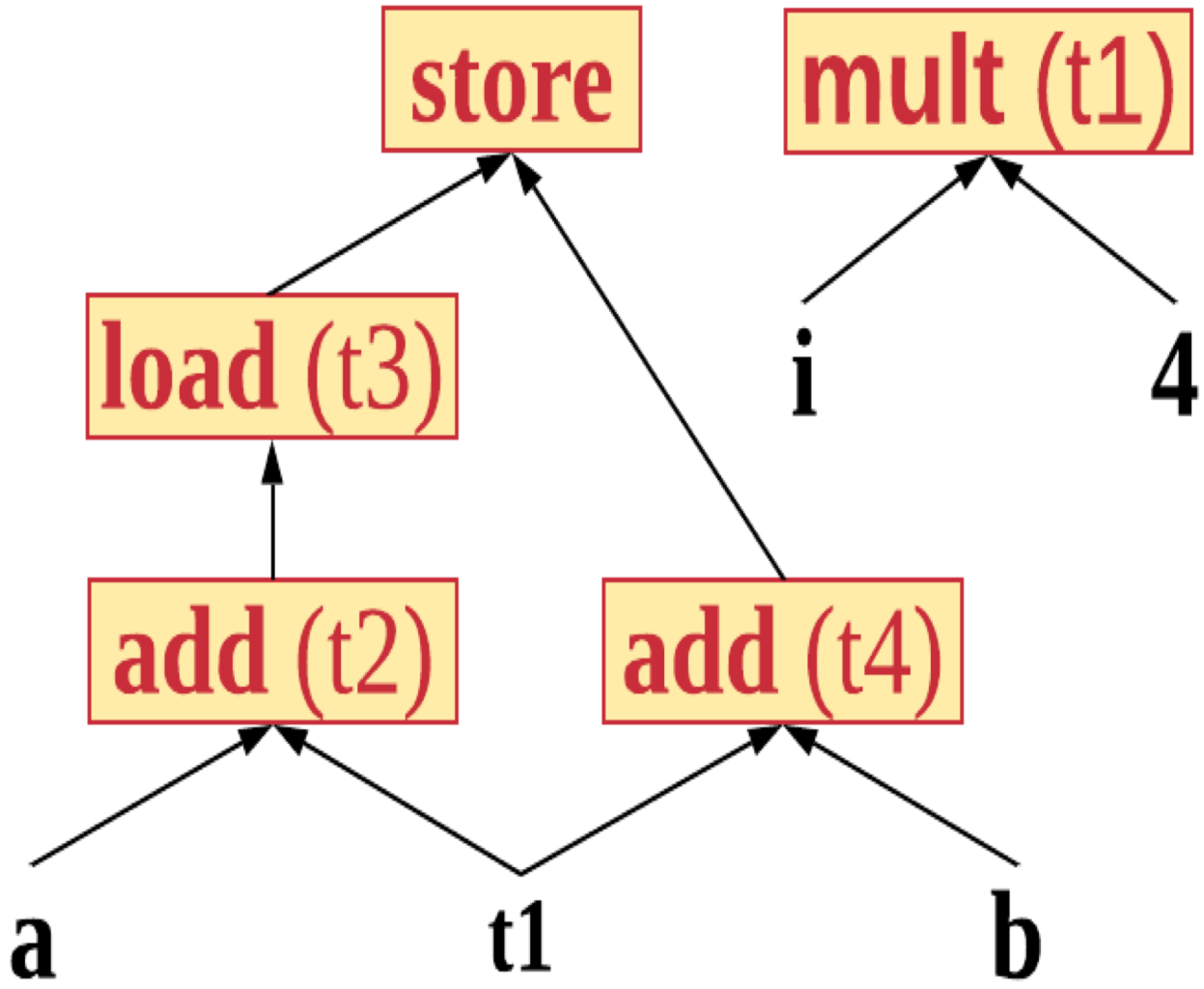


- **Total COST : 14**
  Cheaper than naïve solution!

Dynamic Programming 1 (Bottom-Up)

Node	Best Tile	BestCost
mult(t1)	lshi	2 (lshi)

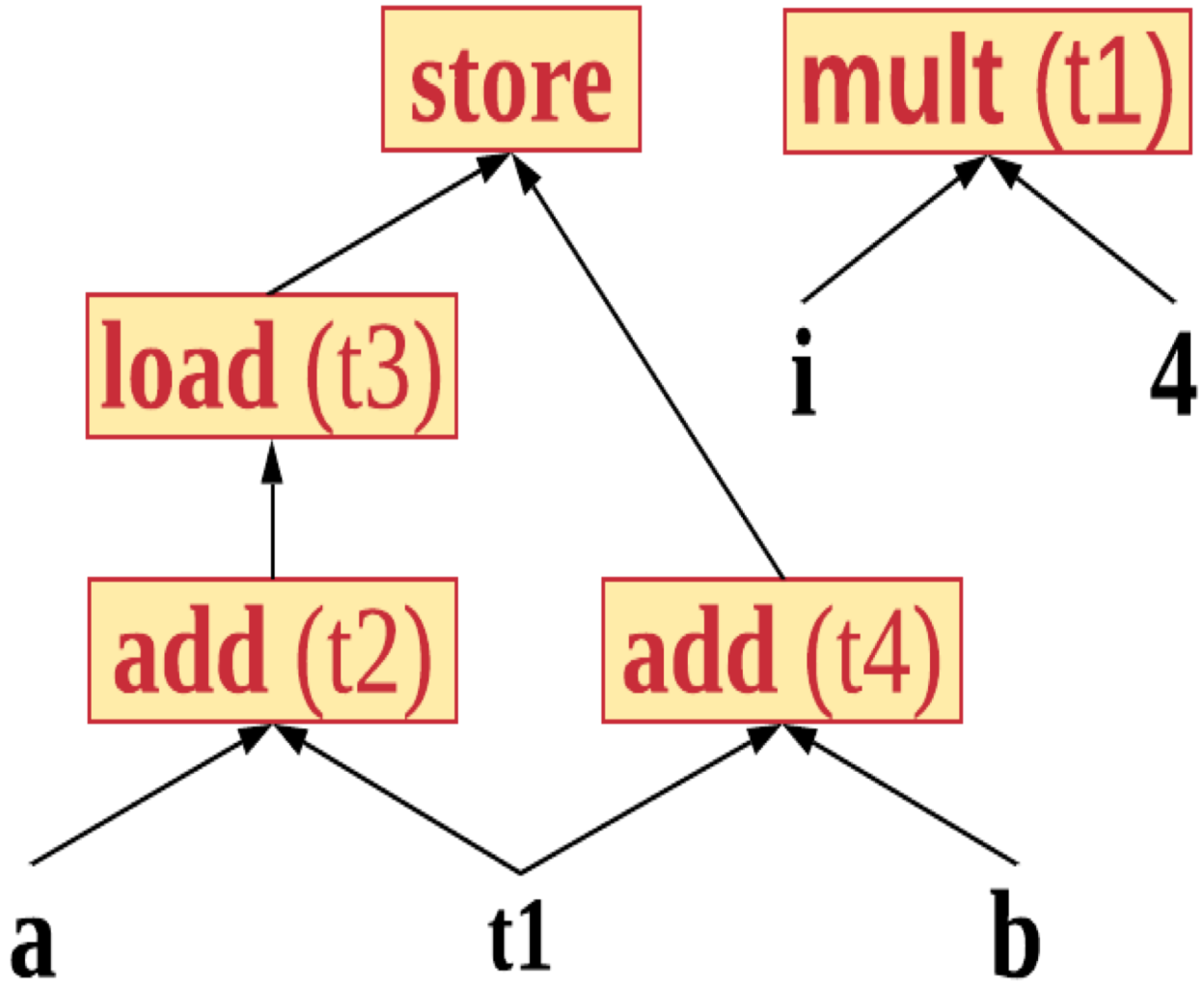
Node	Best Tile	BestCost



Dynamic Programming 2 (Bottom-Up)

Node	Best Tile	BestCost
mult(t1)	lshi	2 (lshi)

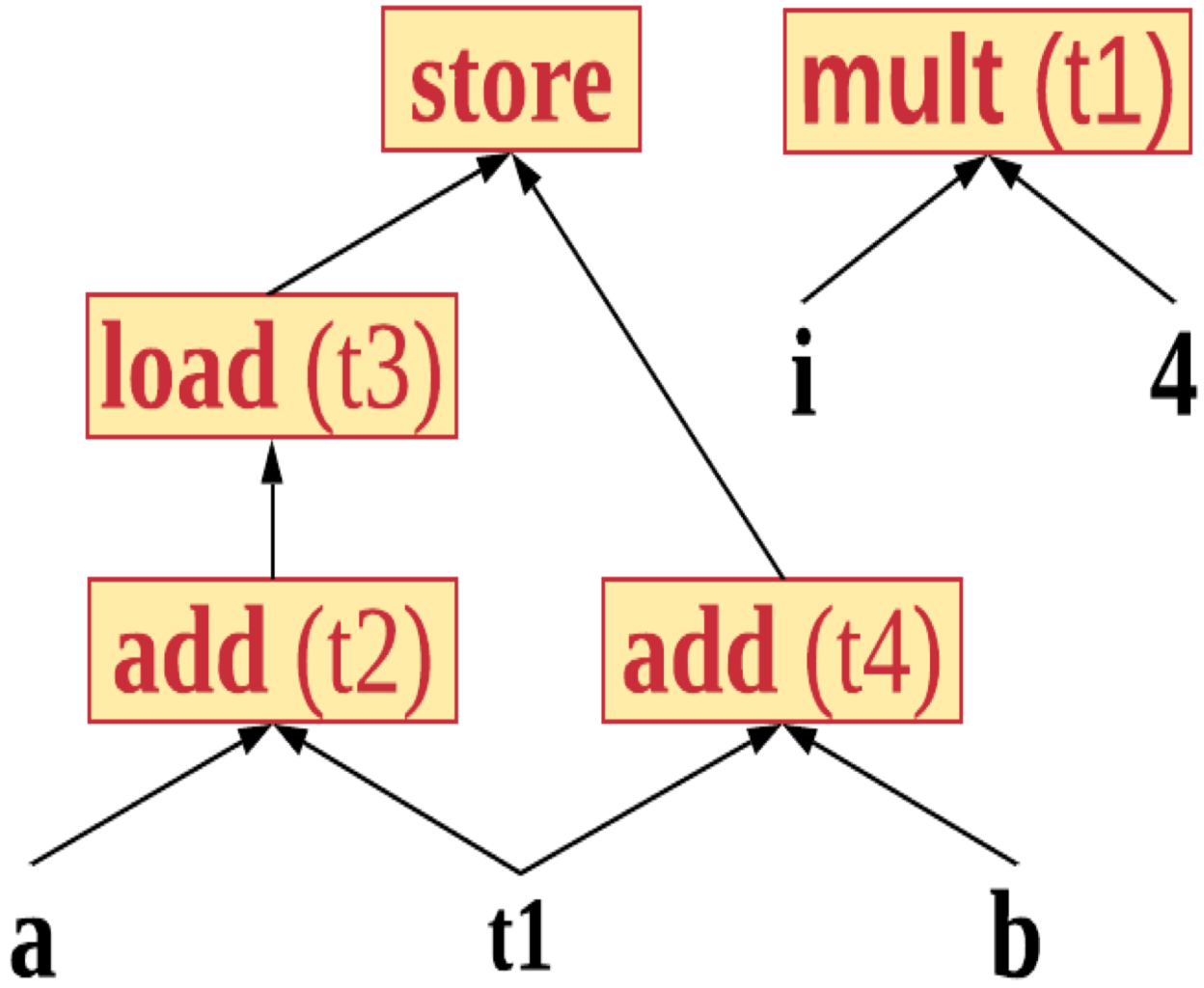
Node	Best Tile	BestCost
Add(t2)	add	2 (add)



Dynamic Programming 3 (Bottom-Up)

Node	Best Tile	BestCost
mult(t1)	lshi	2 (lshi)

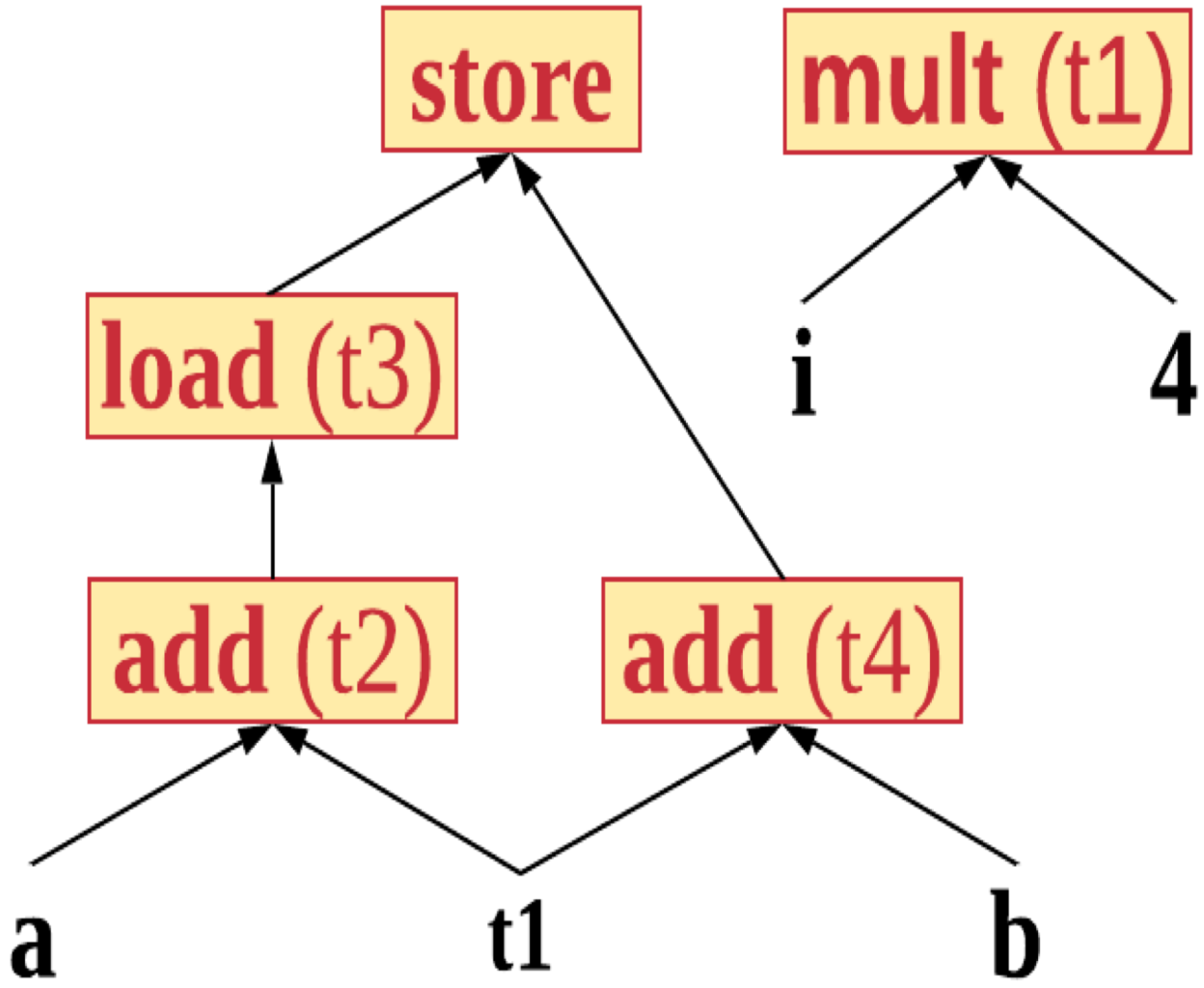
Node	Best Tile	BestCost
Add(t2)	add	2 (add)
Add(t4)	add	2 (add)



Dynamic Programming 4 (Bottom-Up)

Node	Best Tile	BestCost
mult(t1)	lshi	2 (lshi)

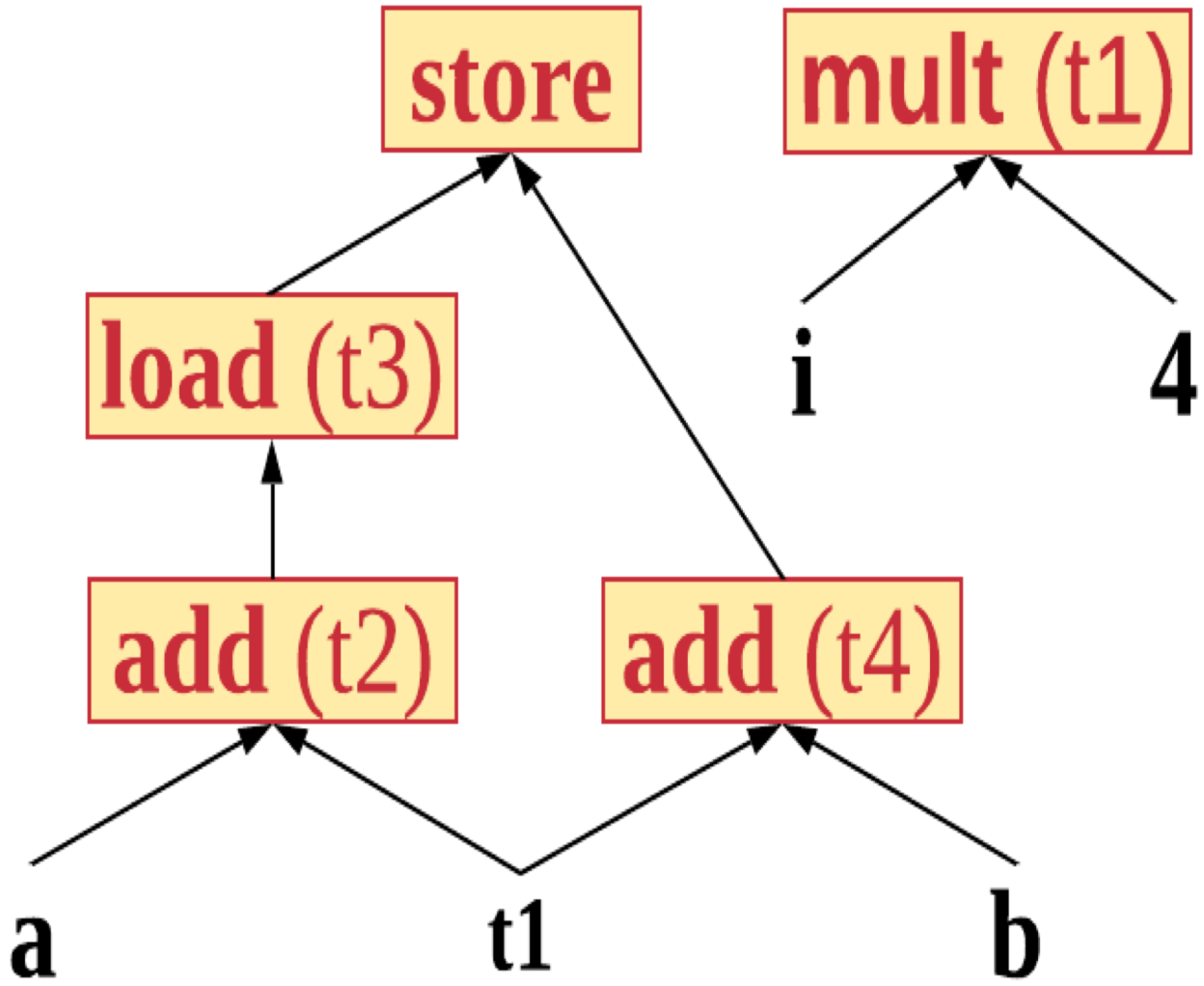
Node	Best Tile	BestCost
Add(t2)	add	2 (add)
Add(t4)	add	2 (add)
Load(t3)	lw	6 (lw) + 2 (t2)



Dynamic Programming 5 (Bottom-Up)

Node	Best Tile	BestCost
mult(t1)	lshi	2 (lshi)

Node	Best Tile	BestCost
Add(t2)	add	2 (add)
Add(t4)	add	2 (add)
Load(t3)	lw	6 (lw) + 2 (t2)
Store (root)	**movmw**	**8 (movmw)** **+ 2 (t2)** **+ 2 (t4)**



Dynamic Programming 6 (Bottom-Up)

Node	Best Tile	BestCost
mult(t1)	lshi	2 (lshi)

Node	Best Tile	BestCost
Add(t2)	add	2 (add)
Add(t4)	add	2 (add)
Load(t3)	lw	6 (lw) + 2 (t2)
Store (root)	**movmw**	**8 (movmw)** **+ 2 (t2)** **+ 2 (t4)**

Resulting cost of instruction selection : **14**

Instruction Selection Result

	Cost
4. Movmw a, b	**8**
3. Add i, b	**2**
2. Add i, a	**2**
1. Lshi 2, i	**2**

- In the worksheet's example,
  Greedy method gives the same result as the optimal Dynamic Programming method.
- However,
  greedy method might not be optimal in other cases.