For CPE performance: Re-association
Check which operations can be precomputed for the next iteration.
R = R*(x*(y*z)) achieves latency*1/3
R = (R*x) * (y*z) achieves *2/3

Performance improvement techniques:

- High-level design—Choose appropriate algorithms and data structures
- Basic Coding Principles: Avoid optimization blockers
    - Eliminate excessive function calls. Move computation out of loops when possible. Consider compromises of program modularity for efficiency.
    - Eliminate unnecessary memory references. Introduce temporary variables to hold intermediate results. Only store in array or global variable after computing final value.
    - Watch for MEMORY ALIASING
- Low-level Optimizations: Structure code to take advantage of hardware
    - Unroll loops to reduce overhead and enable further optimization
    - Find ways to increase instruction-level parallelism by using techniques like multiple accumulators or reassociation.
    - Rewrite conditional operations in a functional style; enables compilation using cmov
        - Long Tempa = a[i], tempb = b[i];
        - Long take1 = a[i]<b[i];
        - min = (take1) ? Tempa : tempb;

Floats are usually NORMALIZED (look at first 1 bit and take it as implied)

Code flow diagram

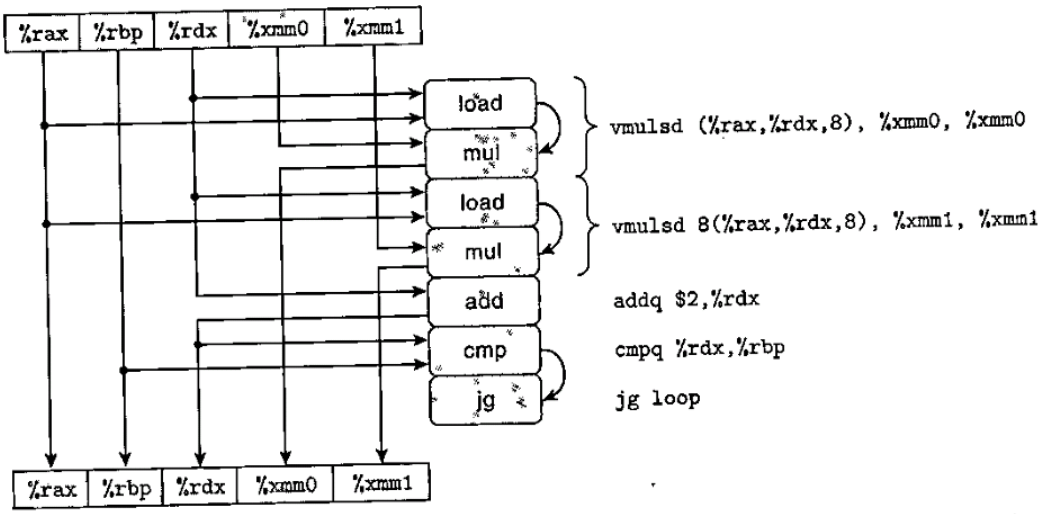
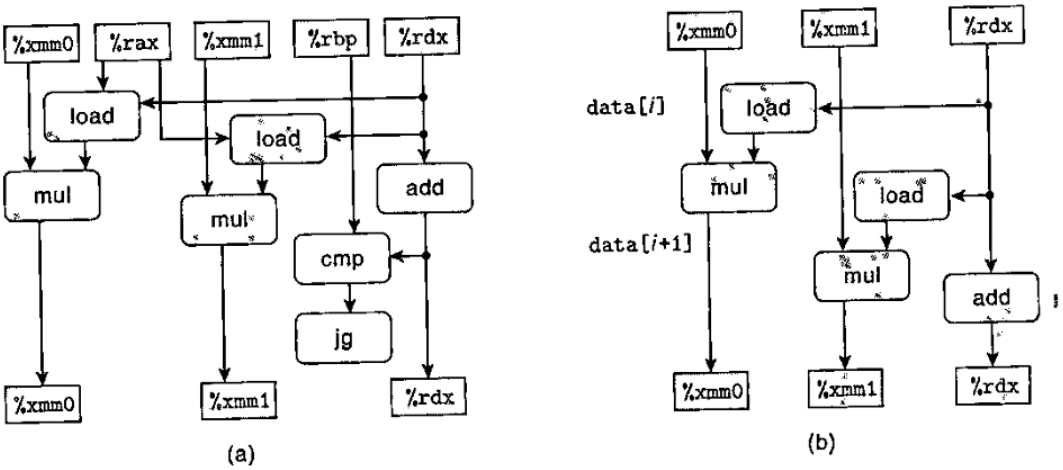
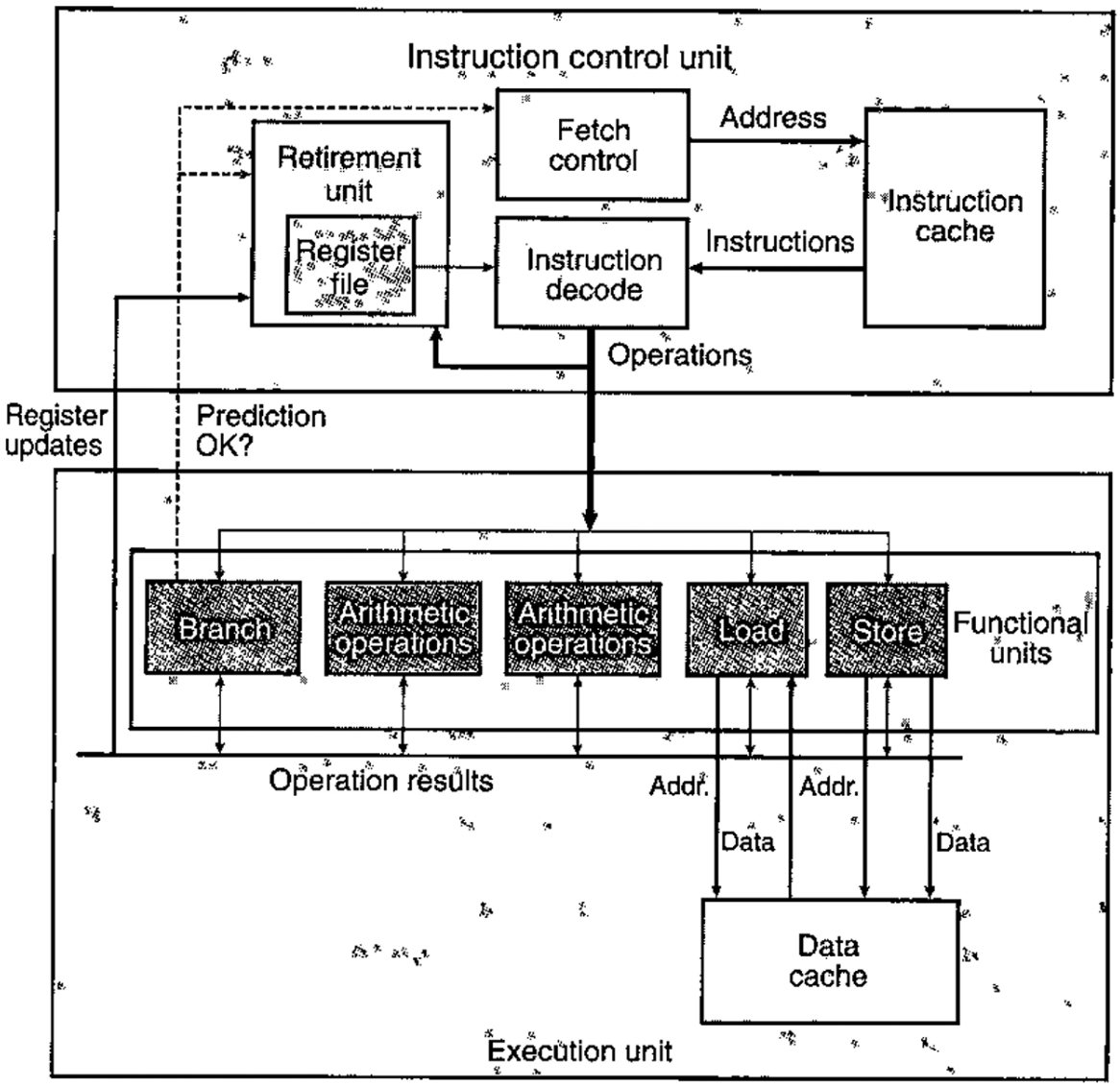


Figure 5.22 Graphical representation of inner-loop code for `combine6`. Each iteration has two `vmulsd` instructions, each of which is translated into a load and a mul operation.



Intel Corei7 Haswell reference machine has 8 functional units. Partial list of each's capabilities:

0. Integer arithmetic (addition bitwise ops and shifting, not mult and div), float mult, integer and float division, branches
1. Integer arithmetic, float add, integer mult, float mult
2. Load, address computation
3. Load, address computation
4. Store
5. Integer arithmetic
6. Integer arithmetic, branches
7. Store address computation (store op requires 2 functional unit)


Instruction control unit
Fetch control
Address
Retirement unit
Register file
Instruction decode
Instructions
Instruction cache
Operations
Register updates
Prediction OK?
Branch
Arithmetic operations
Arithmetic operations
Load
Store
Functional units
Operation results
Addr.
Addr.
Data
Data
Data cache
Execution unit