Announcements

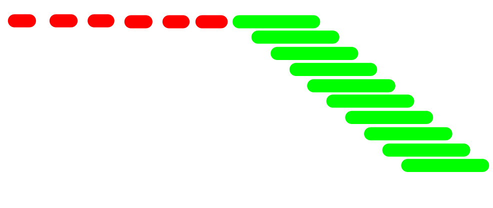
- Reading for this class is Chapter 9
- Reading for Thursday, finish Chapter 9 and start 10
- Project due: 10/3/96

Scalar vs Pipeline vs. Parallel Vector

- Perform the same operations on a collection of data
  - s = startup time
    - a constant that is specific to each type of operation
  - l = suboperations per element
    - steps involved in performing arithmetic
  - n = vector length
- Examples
  - weather code, update the elements of the arrays
  - matrix multiply
    - requires both an add and a multiply for each element
- Design Issues
  - amount of hardware replicated
  - potential performance compared to serial version

Pipeline

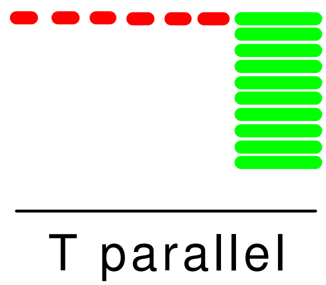
- steps are performed on different operands at once
  - split multiple, add, and others into smaller parts
  - once one step is done, the unit becomes free
- Hardware
  - some amount of execution logic
  - extra hardware to communicate between stages and control
- Performance
  - $T_{pipe} = (s + l)\ t + (n-1)t$
  - for infinite vectors, average time per element is t



T pipeline

Parallel Vector

- Hardware
  - fully replicate ALU
  - need control logic to start each ALU one the right data
- Performance
  - T_{pvp}= n/P(s + l)t
  - average time per vector element is 1/n(s+l)t



Performance Issues

- Vector length
  - how many operands are needed to get "good" performance?
  - $n_{1/2}$ = vector length to get 50% of peak performance
    - also can be viewed at the number of ops that could be performed while filling the pipes
- Memory Bandwidth
  - how many memory refs are generated to vector op?
    - often the real key to performance systems
  - naive case: 3 (op1, op2, dest)
  - improved: 2 (op1, op2)
    - store result into register
  - vector registers: 1 (op1)
    - keep one vector in vector registers
    - result into a register
  - Would caches help?

Cray PvP Machines

Why peak is not equal to sustained

- Cray-1
  - permits chaining of multiply and add
  - peak performance of 160 Mflops (80Mflops times 2)
  - consider vector multiplication
    - only can use multiply, no add needed (80Mflops)
    - need to read two operands, and write one
      - can chain one of these with multiply
      - results in 80/3 or about 23Mflops
- Cray Y-MP
  - three ports to memory
  - peak of 200Mflops
  - can get 70Mlops for vector multiply
    - but requires 500 element vectors to reach 90%
    - startup cost starts to dominate

SIMD Machines

- Replicate functional units
  - ALU, FPU, memory units
- One master controller
  - issues same instruction to each processor
  - mask used to not apply instruction to selected processor

CM-1

- 256 x 256 grid of processors
  - each processor is one bit
  - grouped into 4096 (2^{12}) units of 16 processors each
    - connected by a 12 dimension hypercube
    - also connected to NEWS neighbors
- Processors
  - inputs a & b bits from memory and 1 flag bit
  - output A bit and new flag bit
  - ALU Operators ($2^3 = 8$)

ADD	A← XOR(A,B,F)	Fd←MAJ(A,B,F)
AND	A←AND(A,B,F)	Fd←AND(A,B,F)
OR	A←OR(A,B,F)	Fd←OR(A,B,F)
MOVE	A←OR(A,B,F)	Fd←AND(XOR(A,B),F)
SWAP	A←F	Fd←B