CPUs

- CPU performance
- CPU power consumption.

Elements of CPU performance

- Cycle time.
- CPU pipeline.
- Memory system.

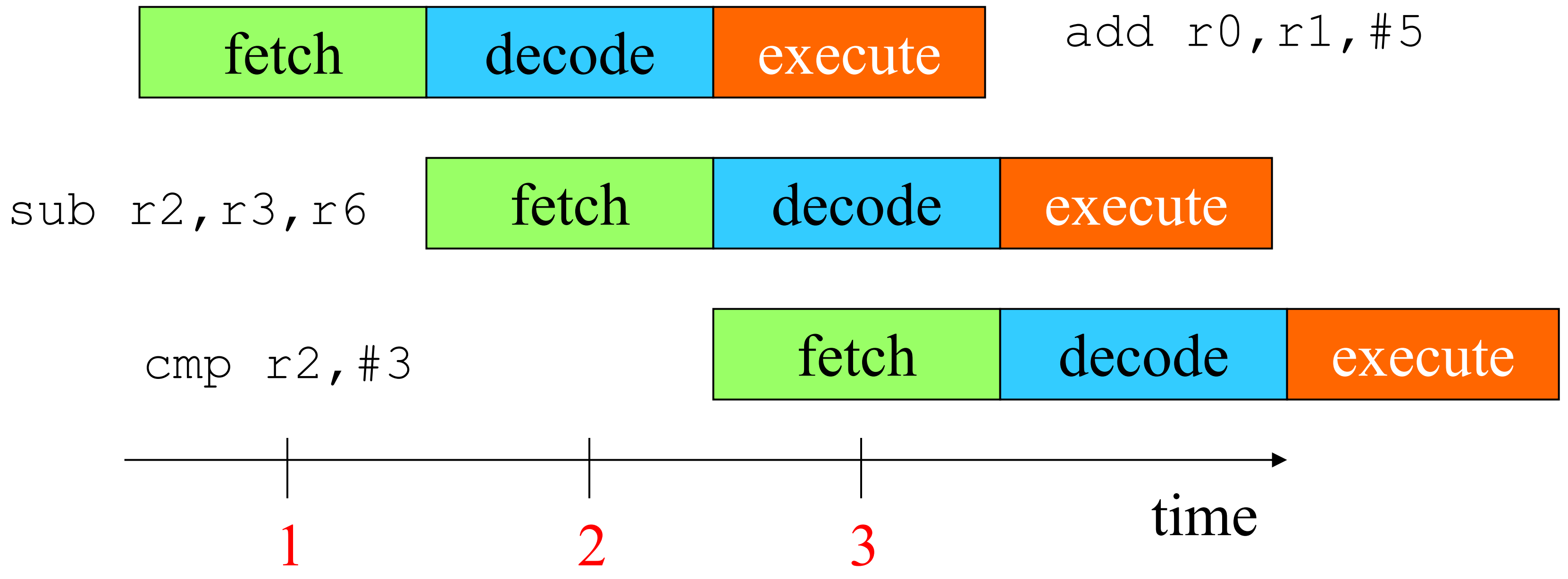
Pipelining

- Several instructions are executed simultaneously at different stages of completion.
- Various conditions can cause pipeline bubbles that reduce utilization:
  - branches;
  - memory system delays;
  - etc.

Pipeline structures

- Both ARM and SHARC have 3-stage pipes:
  - fetch instruction from memory;
  - decode opcode and operands;
  - execute.

ARM pipeline execution



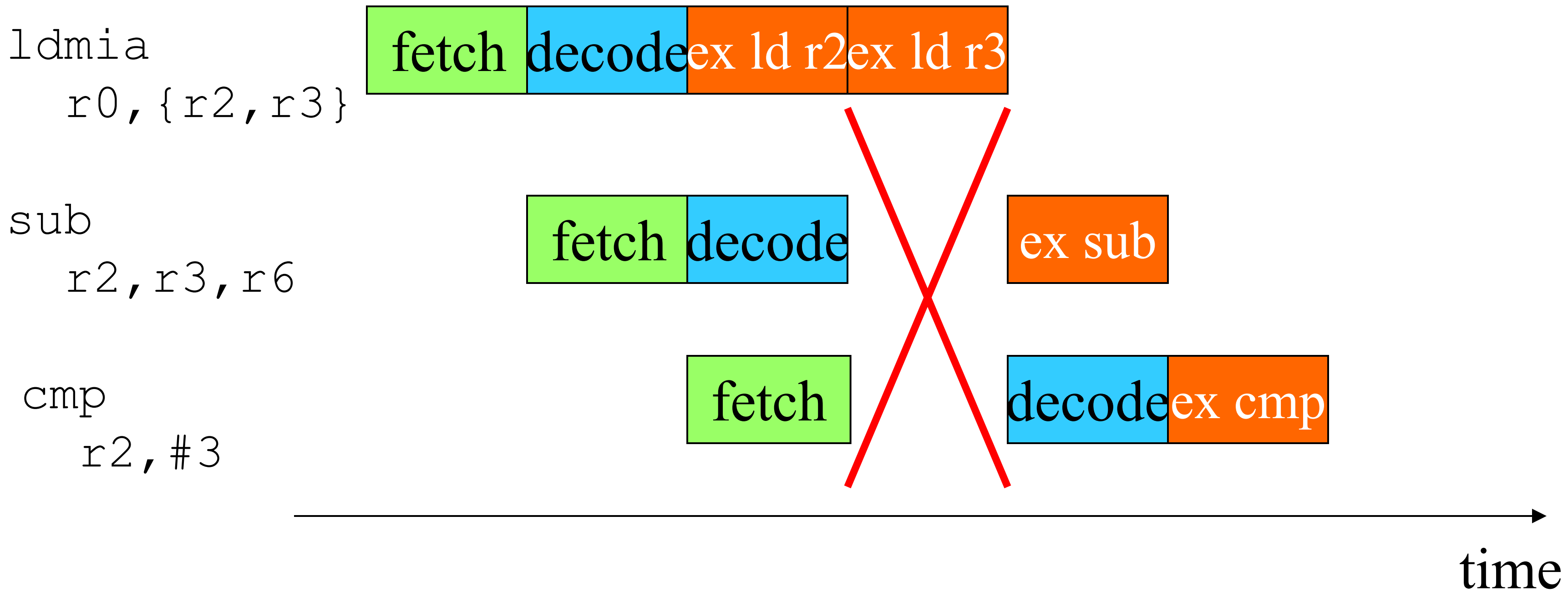
Performance measures

- Latency: time it takes for an instruction to get through the pipeline.
- Throughput: number of instructions executed per time period.
- Pipelining increases throughput without reducing latency.

Pipeline stalls

- If every step cannot be completed in the same amount of time, pipeline stalls.
- Bubbles introduced by stall increase latency, reduce throughput.

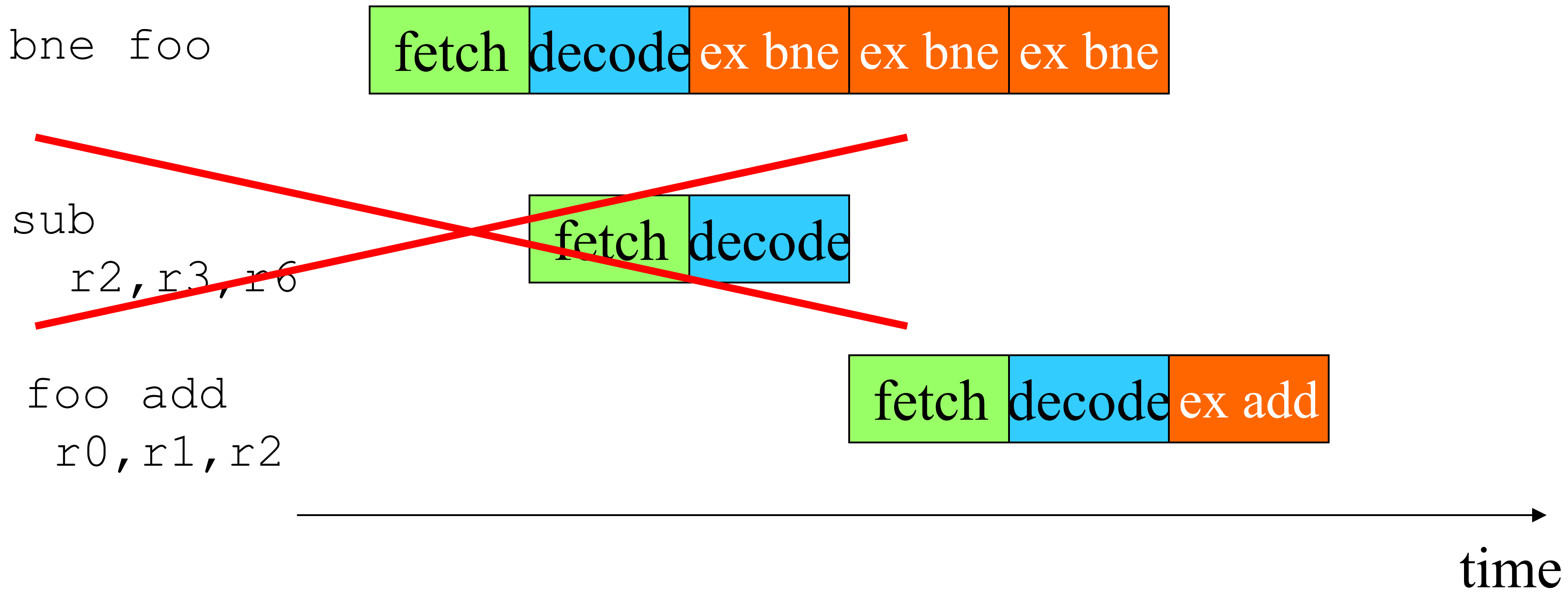
ARM multi-cycle LDMIA instruction



Control stalls

- Branches often introduce stalls (branch penalty).
  - Stall time may depend on whether branch is taken.
- May have to squash instructions that already started executing.
- Don't know what to fetch until condition is evaluated.

ARM pipelined branch



Delayed branch

- To increase pipeline efficiency, delayed branch mechanism requires n instructions after branch always executed whether branch is executed or not.
- SHARC supports delayed and non-delayed branches.
  - Specified by bit in branch instruction.
  - 2 instruction branch delay slot.

Example: SHARC code scheduling

```
L1=5;
DM(I0,M1)=R1;
L8=8;
DM(I8,M9)=R2;
```

- CPU cannot use DAG on cycle just after loading DAG's register.
  - CPU performs NOP between register assign and DM.

Rescheduled SHARC code

```
L1=5;
L8=8;
DM(I0,M1)=R1;
DM(I8,M9)=R2;
```

- Avoids two NOP cycles.

Example: ARM execution time

- Determine execution time of FIR filter:

```
for (i=0; i<N; i++)
  f = f + c[i]*x[i];
```

- Only branch in loop test may take more than one cycle.
  - `BLT loop` takes 1 cycle best case, 3 worst case.

Superscalar execution

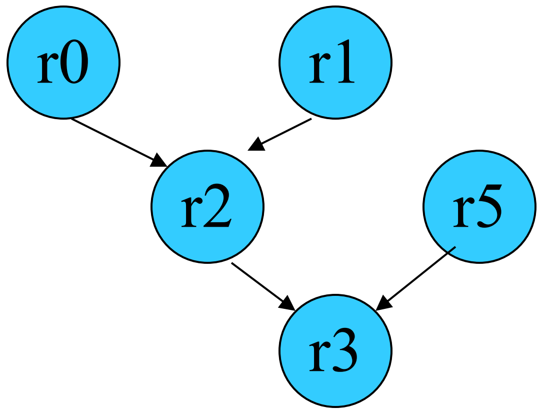
- Superscalar processor can execute several instructions per cycle.
  - Uses multiple pipelined data paths.
- Programs execute faster, but it is harder to determine how much faster.

Data dependencies

- Execution time depends on operands, not just opcode.
- Superscalar CPU checks data dependencies dynamically:

data dependency

```
add r2,r0,r1
add r3,r2,r5
```



Memory system performance

- Caches introduce indeterminacy in execution time.
  - Depends on order of execution.
- Cache miss penalty: added time due to a cache miss.
- Several reasons for a miss: compulsory, conflict, capacity.

CPU power consumption

- Most modern CPUs are designed with power consumption in mind to some degree.
- Power vs. energy:
  - heat depends on power consumption;
  - battery life depends on energy consumption.

CMOS power consumption

- Voltage drops: power consumption proportional to V^2.
- Toggling: more activity means more power.
- Leakage: basic circuit characteristics; can be eliminated by disconnecting power.

CPU power-saving strategies

- Reduce power supply voltage.
- Run at lower clock frequency.
- Disable function units with control signals when not in use.
- Disconnect parts from power supply when not in use.

Power management styles

- Static power management: does not depend on CPU activity.
  - Example: user-activated power-down mode.
- Dynamic power management: based on CPU activity.
  - Example: disabling off function units.

Application: PowerPC 603 energy features

- Provides doze, nap, sleep modes.
- Dynamic power management features:
  - Uses static logic.
  - Can shut down unused execution units.
  - Cache organized into subarrays to minimize amount of active circuitry.

PowerPC 603 activity

- Percentage of time units are idle for SPEC integer/floating-point:

unit	Specint92	Specfp92
D cache	29%	28%
I cache	29%	17%
load/store	35%	17%
fixed-point	38%	76%
floating-point	99%	30%
system register	89%	97%

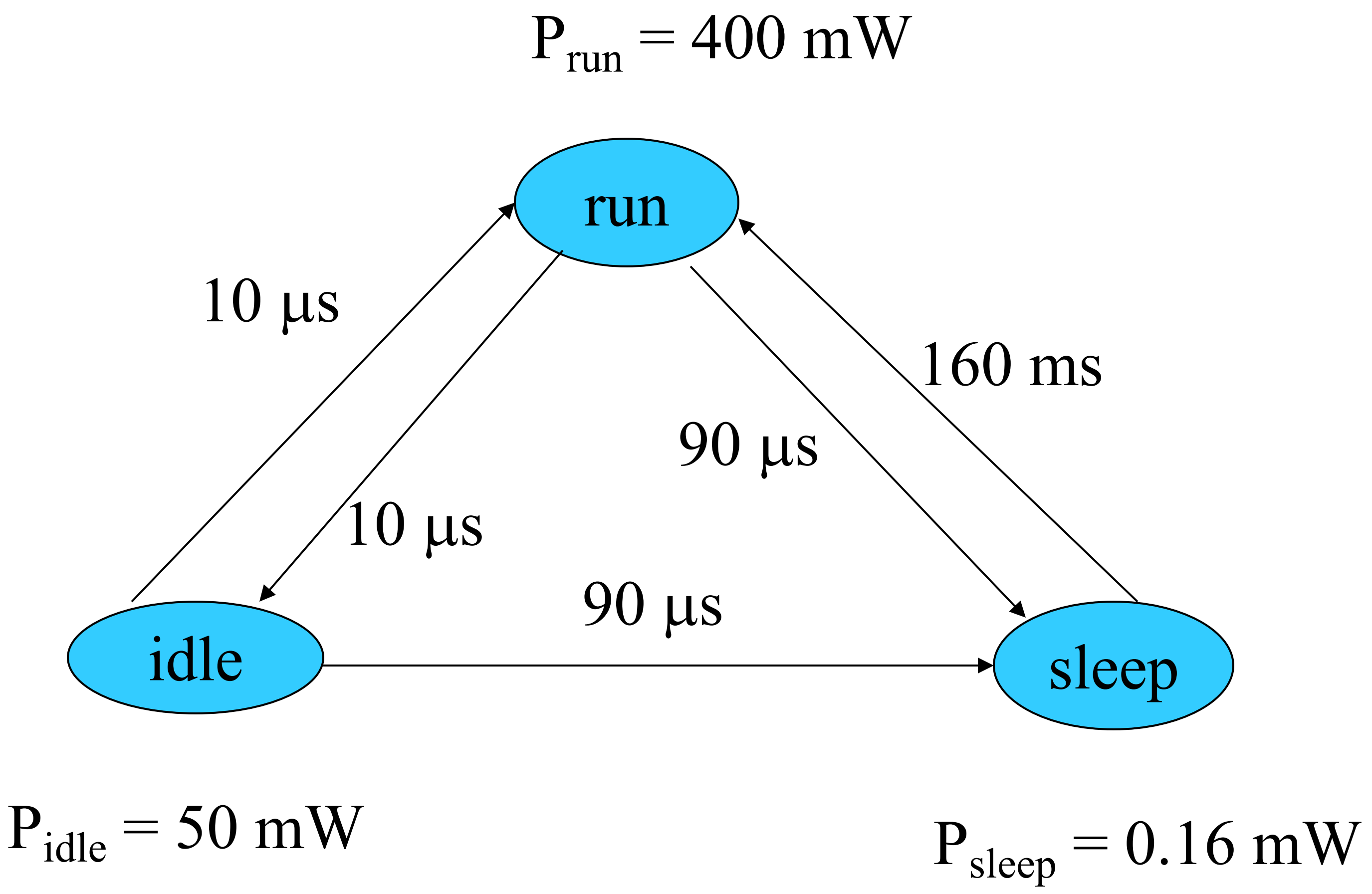
Power-down costs

- Going into a power-down mode costs:
  - time;
  - energy.
- Must determine if going into mode is worthwhile.
- Can model CPU power states with power state machine.

Application: StrongARM SA-1100 power saving

- Processor takes two supplies:
  - VDD is main 3.3V supply.
  - VDDX is 1.5V.
- Three power modes:
  - Run: normal operation.
  - Idle: stops CPU clock, with logic still powered.
  - Sleep: shuts off most of chip activity; 3 steps, each about 30 μs; wakeup takes > 10 ms.

SA-1100 power state machine


P_{run} = 400 mW
run
10 µs
160 ms
90 µs
10 µs
90 µs
idle
sleep
P_{idle} = 50 mW
P_{sleep} = 0.16 mW