Computer Design (0907432)
Homework 1

Solutions

Problems 1-5 4 points each, Problems 6-8 8 points each

1. State the six reasons why latency improvements lag bandwidth improvements.
   1. Moore's Law helps BW more than latency
   2. Distance limits latency
   3. Bandwidth easier to sell ("bigger=better")
   4. Latency helps BW, but not vice versa
   5. Bandwidth hurts latency
   6. Operating System overhead hurts Latency more than Bandwidth

2. Why processor manufacturers have recently shifted to building multicore processors instead of building larger and faster uni-processors?
   Building faster processors have hit the power wall, processors running on higher frequencies dissipate too much power.

   Building larger processors have hit the ILP wall, diminishing returns on larger hardware for more ILP.

3. A program consists of multiplication operations and other operations. The multiplication operations are sped up by a factor of 5. Using Amdahl's law, find out what should be the fraction of the execution time of the multiplication operations (f) to achieve an overall speed up of 4.

   Over all speedup = 1 / ((1-f) + f/s)
   4 = 1 / ((1-f) + f/5)
   4 - 4f + 4f/5 = 1
   3.2 f = 3
   f = 3/3.2
   f = 0.94

4. Calculate FIT and MTTF for a system that has 3 disks (1M hour MTTF per disk), 2 CPUs (2M hour MTTF per CPU), and 4 memory modules (0.5M hour MTTF per module).

   Failure rate = (3 / 1,000,000 + 2 / 2,000,000 + 4 / 500,000) * 10^9
   = 12 * 10^3 FIT

   MTTF = 10^9 / 1.2 * 10^4
   = 83,333 hours

5. For the data shown in the following table, summarize the relative performance of Machine A using one number.

Benchmark	Machine A Execution Time	Reference Machine Execution Time	**Relative Perf**
1	100 sec	500 sec	**500/100 = 5**
2	75 sec	400 sec	**400 / 100 = 5.33**
3	200 sec	450 sec	**400 / 100 = 2.25**
4	150 sec	350 sec	**300 / 150 = 2.33**

Relative Performance = (5 * 5.33 * 2.25 * 2.33)$^{1/4}$

Relative Performance = (137.7)$^{1/4}$

Relative Performance = 3.44

6. Write code sequences to implement D=A*(B-C+A); on the four ISA classes.

Stack:

```
PUSH A
PUSH B
PUSH C
SUB
PUSH A
ADD
MUL
POP D
```

Accumulator:

```
LD B
SUB C
ADD A
MUL A
ST D
```

Register-Memory:

```
LD R1, A
LD R2, B
SUB R2, C
ADD R2, R1
MUL R2, R1
ST D, R2
```

Register-Register:

LD R1, A
LD R2, B
LD R3, C
SUB R2, R3
ADD R2, R1
MUL R2, R1
ST D, R2

7. Show how to implement the addressing modes in Figure B.6 using register, immediate, and register indirect addressing modes.

- Displacement:

```
add r4, 100(r1)      ≡    addi r4, 100
```

- Indexed:

```
add r3, (r1,r2)      ≡    add r1, r2
                          add r3, (r1)
```

- Direct:

```
add r1, (1001)       ≡    li r1, 1001
                          add r1, (r1)
```

- Memory indirect:

```
add r1, @(r3)        ≡    lw r3, (r3)
                          add r1, (r3)
```

- Auto increment:

```
add r1, (r2)+        ≡    add r1, (r2)
                          addi r2, 4
```

- Auto decrement:

```
add r1, -(r2)        ≡    addi r2, -4
                          add r1, (r2)
```

- Scaled:

```
add r1, 100(r2)[r3]  ≡    muli r3, 4
                          add r2, r3
                          addi r2, 100
                          add r1, (r2)
```

8. Using pipeline diagrams, find how many cycles are needed to execute the following code sequence.
    a. When stalls are used to solve hazards and branch instructions are resolved in the Execute stage.
    b. When full forwarding paths are used plus stalls (when needed), one branch delay slot, and branch instructions are resolved in the Decode stage.

```
      lw    r1, 0(r2)
      lw    r3, 4(r2)
      add   r4, r1, r3
      sw    r4, 0(r2)
      beq   r4, r4, skip
      andi  r4, r4, 0
skip: sw    r4, 4(r2)
```

(a) 16 Cycles

```
      lw    r1, 0(r2)    F D E M W
      lw    r3, 4(r2)      F D E M W
      add   r4, r1, r3       F D D D E M W
      sw    r4, 0(r2)          F F F D D D E M W
      beq   r4, r4, skip             F F F D E M W
      andi  r4, r4, 0                      F B B B B
skip: sw    r4, 4(r2)                          F D E M W
```

(b) 12 Cycles

```
      lw    r1, 0(r2)    F D E M W
      lw    r3, 4(r2)      F D E M W
      add   r4, r1, r3       F D D E M W
      sw    r4, 0(r2)          F F D E M W
      beq   r4, r4, skip           F D E M W
      andi  r4, r4, 0                F D E M W
skip: sw    r4, 4(r2)                  F D E M W
```