Homework 1

B00401062 羅文斌

- 1.2.

  a. Performance via Pipelining.

  b. Dependability via Redundancy.

  c. Performance via Prediction.

  d. Make the Common Case Fast.

  e. Hierarchy of Memories.

  f. Performance via Parallelism.

  g. Design for Moore's Law.

  h. Use Abstraction to Simplify Design.

- 1.6.

  a.

  P1: $0.1 * 1 + 0.2 * 2 + 0.5 * 3 + 0.2 * 3 = 2.6$

  P2: $0.1 * 2 + 0.2 * 2 + 0.5 * 2 + 0.2 * 2 = 2.0$

  b. P1: 2.6 * 1.0E6 = 2.6E6. P2: 2.0 * 1.0E6 = 2.0E6.

- 1.10.

  1.

  15cm: Die area: $\pi * 7.5^2/84 = 2.10$. Yield: $1/(1 + 0.020 * 2.10/2)^2 = 0.959$.

  20cm: Die area: $\pi * 10^2/100 = 3.14$. Yield: $1/(1 + 0.031 * 3.14/2)^2 = 0.909$.

  2.

  15cm: $12/(84 * 0.959) = 0.149$

  20cm: $15/(100 * 0.909) = 0.165$

  3.

  15cm: Die area: $\pi * 7.5^2/(84 * 1.1) = 1.91$. Yield: $1/(1 + 0.020 * 1.15 * 1.91/2)^2 = 0.957$.

  20cm: Die area: $\pi * 10^2/(100 * 1.1) = 2.86$. Yield: $1/(1 + 0.031 * 1.15 * 2.86/2)^2 = 0.905$.

  4. Yield = 1/(1 + defect per area unit * die area/2)^2

  $\Rightarrow$ Defect per area unit = 2(yield$^{-0.5}$ -1)/die area

  Yield = 0.92: Defect per area unit = $2(0.92^{-0.5} - 1)/200 = 4.25 * 10^{-4}$ $(1/mm^2)$

  Yield = 0.95: Defect per area unit = $2(0.95^{-0.5} - 1)/200 = 2.60 * 10^{-4}$ $(1/mm^2)$

- 1.14.
  1. Let x be the CPI of FP after improvement.
     $50 * 1 + 110 * 1 + 80 * 4 + 16 * 2 = 2(50 * x + 110 * 1 + 80 * 4 + 16 * 2) \Rightarrow x = -4.12$. Therefore, it is *not* possible.
  2. Let x be the CPI of L/S after improvement.
     $50 * 1 + 110 * 1 + 80 * 4 + 16 * 2 = 2(50 * 1 + 110 * 1 + 80 * x + 16 * 2) \Rightarrow x = 0.8$. Therefore, CPI of L/S should be reduced by 80%.
  3. $(50 * 0.6 + 110 * 0.6 + 80 * 2.8 + 16 * 1.4)/(50 * 1 + 110 * 1 + 80 * 4 + 16 * 2) = 0.669$. Execution time before improvement: $(50 * 1 + 110 * 1 + 80 * 4 + 16 * 2) * 1E6/2E9 = 0.256$. Execution time after improvement: $0.256 * 0.669 = 0.171$.
- 2.4. `B[g] = A[f] + A[f+1]`
- 2.5.

```
sll $t0, $s0, 2
add $t0, $s6, $t0
sll $t1, $s1, 2
add $t1, $s7, $s1
lw $s0, 0($t0)
lw $t0, 4($t0)
add $t0, $t0, $s0
sw $t0, 0($t1)
```

- 2.18.
  1. op: 8 bits; rs,rt,rd: 7 bits.
  2. op: 8 bits; rs,rt: 7 bits.
  3. More opcodes and registers inflate the total number of bits in a single instruction (if the number of bits used to represent an address is not compromised). However, more opcodes allow assembly programs to complete the same task with fewer instructions, and more registers reduces the chances of register spills, thus, decreases the need of using sw and lw to store and restore registers.

- 2.27

```
    addi $t0, $0, 0
    j TEST1
LOOP1:
    addi $t1, $0, 0
    j TEST2
LOOP2:
    add $t2, $t0, $t1
    sll $t3, $t1, 4
    add $t3, $t3, $s2
    sw $t2, 0($t3)
    addi $t1, $t1, 1
TEST2:
    slt $t2, $t1, $s1
    bne $t2, $0, LOOP2
    addi $t0, $t0, 1
TEST1:
    slt $t2, $t0, $s0
    bne $t2, $0, LOOP1
```

- 2.28. 14 instructions. The order of instructions: line 1, 2, (TEST1, LOOP1, 10-11, LOOP2, TEST2) * 10 times, 13, 14. Total number of instructions: 144.
- 2.43.

```
TRY:
    addi $t1, $0, 1
    ll $t0, 0($a0)
    bne $t0, $0, TRY
    sc $t1, 0($a0)
    beq $t1, $0, TRY
    lw $t2, 0($a1)
    slt $t3, $t2, $a2
    beq $t3, $0, UNLOCK
    sw $a2, 0($a1)
UNLOCK:
    sw $0, 0($a1)
```

- 2.44.

```
TRY:
    ll $t2, 0($a1)
    slt $t3, $t2, $a2
    beq $t3, $0, EXIT
    addi $t2, $a2, 0
    sc $t2, 0($a1)
    beq $t2, $0, TRY
EXIT:
```

- 2.45.

  Both processes cannot perform the update unless they both check and pass sc. If the lock ($a0) is 1, meaning that some process is occupying the lock, then both processes should retry until the lock is released. If at some time, the lock is finally released (lock set to 0), and both processes enters sc in the same cycle, only one process can store 1 to the lock successfully, and the other process would detect the change of lock status and fail to retrieve the lock. This mechanism makes sure at most one process is allowed to enter the critical section at a time.