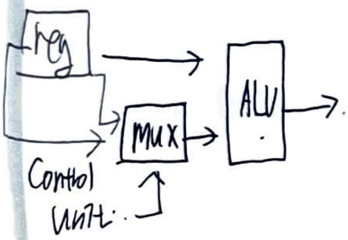
컴퓨터구조
2023-1학기
과제 2
<이성원 교수님>

학과: 컴퓨터정보공학부
학번: 20192021073
이름: 이윤비
마감date: 23/04/12

4.1) Figure 2를 기반으로 Signals 값들을 표로 정리하면 다음과 같습니다.

4.1.1)

RegWrite	MemRead	ALUMux	MemWrite	ALUop	RegMux	(Branch) B.
0	0	1 (Immediate)	1	ADD	X	0



구조를 따르고 있으므로, ALUMux는 ALU 입력에서 Mux를 제어하는 신호이며, 0(Reg)은 regfile의 출력, 1 (immediate value)는 명령어에서 ALU의 2번째 입력으로 그 값을 선택합니다.

RegMux는 ~~데이터~~ regfile에 대한 데이터 입력에서 Mux를 제어하는 제어 신호이며, 0(ALU)은 ALU의 출력을 선택하고, 1 (Mem)은 메모리의 출력을 선택합니다. * Branch에서 X 값은 Don't care를 의미합니다.

4.1.2) branch Add 와 registered writeport 제외한 모든 source.

4.1.3) 사용되지 않는 outputs : Branch Add, registered writeport.
그 외 모든 units는 output을 만들어 냅니다.

4.4.3) 4.4.2 에서와 같이 시간을 구하되, (conditional) PC-relative branch 만 지원합니다.

명령어에 접근하기 위한 << 2 작업, 이후 I-Mem, Register, Mux, ALU가 차례로 계산됩니다.
그리고 이 계산은 PCsrc condition을 계산하기 위해서 이고, 이 path의 latencies는 표에 따라

20 + 200 + 20 + 90 + 20 + 90 = 420ps 인 것을 알 수 있습니다.

4.6.5) 4.6.4 의 경우에 추가조건을 설정한 것으로, RegDst 신호가 0일때 Jump control 신호가 0이 되는 fault가 있는 test를 진행합니다.

Jump가 0일때 fault, 따라서 Jump신호가 1일때 instruction이 필요합니다.
근데, Jump instruction 일 경우 RegDst 신호가 레지스터에 기록되지 않기 때문에
don't care "X" 이고, 따라서 RegDst는 0으로 설정되도, 안되도 가능합니다.
따라서 fault case 잡아 낼 신뢰할만한 test를 할 수 없습니다.

4.7.1)

무관한 →	r0	r1	r2	r3	r4	r5	r6	r8	r12	r31
	0	-1	2	-3	-4	10	6	8	2	-16

0011000100000000000000 1010000 — Jump's shift left

sign-extended > 00000000000000000000000000010100

4.8.2) pipeline 상에서 lw instruction의 총 지연시간은 얼마이고, non-pipelined에서는 얼마인가요

pipe-line 이 아닐때는 4.8에서 제시한 표순으로 1 cycle 진행한다고 생각하면돼.

non pipe-lined 진행에서는 IF→ID→EX→MEM→WB.

pipe-lined 〃 IF→ID→IF→IF→EX→MEM→WB 과 같은 진행에서.

각각 1. → pipeline = 1750 ps (250ps + 350ps + 200ps + 250ps + 150ps + 300ps + 200ps)

non-pipeline = 1250 ps (250ps + 350ps + 150ps + 300ps + 200ps)

1.2.a) 1.2 문제에서 제시한 8가지 아이디어 중. Assembly lines in automobile manufacturing 은

Performance via Pipeline 과 매칭될 수 있습니다.

1.4.b) 각 기본 색상에 대해 8비트를 사용하는 색상 디스플레이를 가정합니다. 픽셀당 (R, G, B)

및 frame size는 1280×1024 입니다. 1280×1240 pixels = 1,310,720 pixels 이고, 각 색상 R, G, B 3개이므로 3×1,310,720 = 3,932,160 bytes/frame. 이값을 통해 프레임이 100M bits/s 네트워크를 전송하는데 걸리는 최소시간은

$$\left(\frac{\frac{3,932,160 \text{ bytes}}{1 \text{ frame}} \times \frac{8 \text{ bits}}{1 \text{ bytes}}}{\frac{100M \text{ bits}}{1 \text{ second}}} \right) = 0.31 \text{ seconds}$$

(1 frame = 100M bits/second)

1.6 b) 같은 instruction의 다른 2가지 구현을 고려합니다.

명령어는 CPI (class A, B, C, D)에 따른 4개의 클래스로 나눌 수 있습니다. clock rate = 2.5GHz이고 CPI가 1,2,3,3,3 인 P1 과 clock rate가 3GHz이고 CPI가 2,2,2,2,2 인 P2입니다. dynamic instruction 수가 1.0E6인 프로그램은 class A 10%, class B 20%, class C 50%, class D 20%로 나눈다면 어떤 구현이 빠를까요.

(b) clock cycle (P1) $= 10^5 \times 1 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 3 + 2 \times 10^5 \times 3 = 26 \times 10^5$

clock cycle (P2) $= 10^5 \times 2 + 2 \times 10^5 \times 2 + 5 \times 10^5 \times 2 + 2 \times 10^5 \times 2 = 20 \times 10^5$

1.8.1) Pentium 4 processor의 processor의 clockrate 3.6 GHz, 전압은 1.25V였습니다. 평균적으로 정적 전력 10W와 동적전력 90W를 소비했다고 가정합니다. Core i5 Ivy Bridge는 clock rate 3.4GHz, 전압 0.9V였습니다. 평균적으로 정적 전력 30W, 동적전력 40W를 소비했다고 가정합니다. 이때 위 2개 각 processor에 대해 평균 capacity loads를 찾으면

$C = 2 \times DP/(V^2 * F)$ 식에 의하여 Pentium 4는 $C = 3.2E-8F$.
Core i5는 $C = 2.9E-8F$ 입니다.

1.10.4) 15cm 직경의 wafer의 cost는 12, 84 dies를 포함하며 0.020 defects/cm², 20cm 직경의 wafer는 15의 cost, 100 dies를 포함하며 0.031 defects/cm². yield가 0.92에서 0.95가 된다고 가정할 때, 200mm²의 die area에 위 내용에서 단위 면적당 결점 수를 찾으면?

단위면적(0.92) 당 defects 수 $= (1-y^{.5})/(y^{.5} * die\_area/2)$

여기서 y=0.95, die_area=2 적용하여 계산.

$= (1-0.92^{.5})/(0.92^{.5} * 2/2)$

$= 0.043$ defects/cm².

단위면적(0.95)당 〃 = (y=0.92, die_area=5)

$= (1-0.95^{.5})/(0.95^{.5} * 2/2)$

$= 0.026$ defects/cm².

1.11.3) 문제에서 제시된 조건에서 CPI 영향을 미치지 않고 benchmark에 10% 증가되었을 명령어의 수가 많아짐에 따라 CPU시간이 증가하는데, 이를 계산하면 다음과 같습니다.

CPU time = No. instr. × CPI / clock rate, 그리고 만약 CPI와 clock rate가 변하지 않는다면 CPU시간은 instruction이 증가된 10%에 비례해서 증가함을 유추할 수 있습니다.

1.11.9) CPU time = No. instr × CPI / clock rate

∴ No. instr = CPU time × clock rate / CPI

$= 960 \times 0.9 \times 4 \times 10^9 / 1.61 = 2146 \times 10^9$

1.12.3) MIPS의 서로 다른 프로세서의 성능을 비교합니다.

MIPS = clock rate $\times 10^{-6}$ / CPI. → MIPS(P2) = $4 \times 10^9 \times 10^{-6} / 0.9 = 4.44 \times 10^3$

MIPS(P2) = $3 \times 10^9 \times 10^{-6} / 0.75 = 4.0 \times 10^3$.

MIPS(P1) > MIPS(P2), performance (P1) < performance (P2)

1.11.11) Clock rate = No. Instr. × CPI / CPU time.

Clock $rate_{new}$ = No. Instr. × 0.85 × CPI / 0.80 CPU time = 0.85/0.80,

Clock rate old = 3.18 GHz.

1.14.3) Clock cycle을 계산하는 것은 각 instruction의 no.of 곱해서 CPI 시간을 곱한 값들을 합한 것입니다. 따라서 Clock cycles = $CPI_{fp} \times$ No. FP instr. + $CPI_{기타} \times$ No. INT instr + $CPI_{L/S} \times$ No. L/S instr, + $CPI_{branch} \times$ No. branch instr 이고, T_{CPU}는 위 값을 이용하여

T_{CPU} = clock cycles / clock rate = clock cycles / 2×10^9

$CPI_{기타} = 0.6 \times 1 = 0.6$ / $CPI_{fp} = 0.6 \times 1 = 0.6$ / $CPI_{L/S} = 0.7 \times 4 = 2.8$ / $CPI_{branch} = 0.7 \times 2 = 1.4$

위 모든 값을 적용하면 최종적으로

T_{CPU} (before) = 0.256 s

T_{CPU} (after) = 0.1171 s 로 개선된 것을 알 수 있습니다.