Computer Architecture HW 3

2013210111 남세현

4.1 Consider the following instruction:

Instruction: AND Rd, Rs, Rt

Interpretation: Reg[Rd] = Reg[Rs] AND Reg[Rt]

4.1.1 What are the values of control signals generated by the control in Figure 4.2 for the above instruction?

Answer:

RegWrite = 1

MemRead = 0

MemWirte = 0

ALU Operation = AND(0000, according to textbook 259 page)

Branch = 0

4.1.2 Which resources (blocks) perform a useful function for this instruction?

Answer: Without Data memory, All of Blocks.

4.1.3 Which resources (blocks) produce outputs, but their outputs are not used for this instruction? Which resources produce no outputs for this instruction?

Answer:

Not Used Outputs : 잘 모르겠습니다. 없는 것 같습니다.

No Outputs : Data Memory

4.2 The basic single-cycle MIPS implementation in Fugure 4.2 can only implement some instructions. New instructions can be added to an existing Instruction Set Architecture(ISA), but the decision whether or not to do that depends, among other things, on the cost and complexity the proposed addition introduces into the processor datapath and control. The first three problems in this exercise refer to the new instruction:

Instruction: LWI Rt, Rd(RS)

Interpretation: Reg[RT] = Mem[Reg[Rd]+[Reg[Rs]]

4.2.1 Which existing blocks (if any) can be used for this instruction?

Answer:

Registers, ALU, Data Memory.

4.2.2 Which new functional blocks (if any) do we need for this instruction?

Answer

Don't need.

4.2.3 What new signals do w need (if any) from the control unit to support this instruction?

Answer

Memory to Register

4.8 In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

alu	beq	lw	sw
45%	20%	20%	15%

4.8.1 What is the clock cycle time in a pipelined and non-pipelined processor?

Answer

Pipelining to 5 stages reduces the cycle time to the length of the longest stage. So, 350ps.

논 파이프라인드 프로세서에서는 한 스테이지를 실행하기 위해선 전 스테이지가 끝날 때 까지 기다려야 하므로, 총 스테이지에 걸리는 시간의 합과 같다. 그러므로 1250ps.

4.8.2 What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

Answer

LW need the stages IF, ID, MEM and WB.

Pipelined : max(250,350,300,200) = 350ps.

Non pipelined : sum(250,350,300,200) = 1100ps.

4.8.3 If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

Answer

ID. Becuase ID has the biggest latency. On pipelined processor, the cyecle time is depend on the biggest latency of the stages.

4.8.4 Assuming there are no stalls or hazards, what is the utilization of the data memory?

Answer

Lw and sw use data memory, so 20 + 15 = 35%

4.8.5 Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

ALU and LW use Register's write-register port. 45 + 20 = 65%.

4.8.6 Instaed of a single-cycle organization, we can use a multi-cycle organization where each instruction takes multiple cycles but one instruction finisheds before another is fetched. In this organization, an instruction only goes through stages it actually need (e.g., ST only takes 4 cycles because it dose not need the WB stage). Compare clock cycle times and execution times with single-cycle, multi-cycle, and pipelined organization.

Assume there are N of the instruction.

Single-cycle should go all the stages. So the clock cycle time is sum of the stages' latency.
Single-cycle need N * clock cycle times because single-cycle can't reduce the stages.

Multi-cycle's clock cycle time is dynamic on the stages' latency.

But pipelined's clock cycle time is static, maximum of the stages' latencies.

Multi-cycle can reduce the stages. Pipelined don't have to wait until previous instruction finished.

Multi-cycle need compact times of instructions for execution times. No waste stage.

Pipelined need 4 + N * cycle clock time for execution times.

케이스 바이 케이스로 다르기 때문에 멀티 사이클과 파이프라인드의 실행시간은 비교하기 어렵다.

4.10 in this exercise, we examine how resource hazards, control hazards, and Instruction Set Architectrue(ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
        sw      r16, 12(r6)
        lw      r16, 8(r6)
        beq     r5, r4, LABEL # assume r5 != r4
        add     r5, r1, r4
        slt     r5, r15, r4
```

Assume that individual pipeline stages have the following latencies:

IF	ID	EX	MEM	WB
200ps	120ps	150ps	190ps	100ps

4.10.1 For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we only have on memory (for both instructions and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forware progress, this hazard must always be resolved in favor of the instruction that accesses data. What is the total execution time of this instruction sequence in the 5-stage pipelin that only has one memory? We have seen that data hazards can be eliminiated by adding nops to the code. Can you do the same with this structural hazard? Why?

Answer

MEM과 IF를 동시에 사용하지 못하는 것이기 때문에, 그런 상황에서 stall을 하면 될 것 같습니다.

SW와 LW MEM을 사용할 때가 바로 그 상황인데, 그때 IF를 실행해야하는 BEQ instruction을 두 클럭 뒤에 실행하면 됩니다.

sw	r16, 12(r6)	IF	ID	EX	**MEM**	WB						
lw	r16, 8(r6)		IF	ID	EX	**MEM**	WB					
beq	r5, r4, LABEL			IF	ID	EX	MEM	WB				
add	r5, r1, r4						IF	ID	EX	MEM	WB	
slt	r5, r15, r4							IF	ID	EX	MEM	WB

총 11clock이며, 한 클럭은 200ps이므로 total execution time = 2200ps 입니다.

Nops도 instruction이라 fetching을 해야하니까 큰 도움이 안될 것 같습니다.

4.10.2 For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we change load/store instrucitons to use a register (witchout an offset) as the address, these instructions no longer need to use the ALU. As a result, MEM and EX stages can be overlapped and the pipeline has only 4 stages. Change this code to accommodate this changed ISA. Assuming this change dose not affect clock cycle time, what speedup is achieved in this instruction sequence?

Answer

Stage가 5개였을 때엔 9 clock cycles이 필요하다면, 4개가 되면 8clock cycles이 필요합니다.

9/8 배 더 빨라졌다고 볼 수 있습니다.

4.10.3 Assuming stall-on-branch and no delay slots, what speedup is achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

Answer

Stall-on-branch라고 했으니, EX 스테이지에서 결정된다고 하면 총 2 clock cycles를 stall해야 하고, ID 스테이지에서 결정된다고 하면 1 clock cycle를 stall해야 합니다.

전자의 경우 11 clock cycles, 후자는 10 clock cycles이므로 1.1배 더 빨라지겠습니다.

4.10.4 Given these pipeline stage latencies, repeat the speedup calculcation from 4.10.2, but take into account the (possible) change in clock cycle time. When EX and MEM are done in a single stage, most of their work can be done in parallel. As a result, the resulting EX/MEM stage has a latency that is the larger of the original two, plus 20 ps needed for the work that could not be done in parallel.

Answer

Clock cycle time이 바뀌게 됩니다. 원래는 IF 스테이지의 latency인 200ps인데, MEM + 20ps인 210ps가 가장 오래 걸리므로

기존 5스테이지 일 땐 9 * 200ps = 1800ps

4스테이지에 210ps일땐 8 * 210ps = 1680ps

1800/1680 = 1.07배 speedup입니다.

4.10.5 Given these pipeline stage latencies, repeat the speedup calculation from 4.10.3, taking into account the (possible) change in clock cycle time. Assume that the latency ID stage increases by 50% and the latency of the EX stage decreases by 10ps when branch outcome resolution is moved from EX to ID.

Answer

어차피 pipelined 에서는 최장시간 latency가 중요하므로, EX가 10ps 줄어든 것은 큰 영향을 미치지 않고, ID가 50% 증가한 것은 중요하게 작용합니다.

이로서 clock cycle time은 200 * 150% = 300ps가 되었고,

바뀌기 전에는 200ps에 11cycle 이었으므로 2200ps,

바뀐 후에는 300ps * 10cycle = 3000ps

Speedup = 2200 / 3000 = 0.73

4.10.6 Assuming stall-on-branch and no delay slots, what is the new clock cycle time and execution time of this instruction sequence if beq address computation is moved to the MEM stage? What is the speedup from this change? Assume that the latency of the EX stage is reduced by 20 ps and the latency of the MEM stage is unchanged when branch outcome resolution is moved from EX to MEM.

Answer

최장 latency는 IF인데 EX의 latency 감소는 큰 영향을 미치지 않습니다.

바뀌기 전에는 전 문제풀이에서 200ps * 11cycle = 2200ps라는 것을 알수 있습니다.

바뀐 내용을 적용해보면, Beq가 EX가 아닌 MEM에서 결정된다고 하면 총 stall은 3사이클 일어납니다. 그러면 총 실행 사이클은 12cycle이 필요합니다. Latency는 바뀐 게 없으므로

Speedup = 11/12.

4.11 Consider the following loop.

```
loop:lw   r1,0(r1)
     and r1,r1,r2
     lw   r1,0(r1)
     lw   r1,0(r1)
     beq r1,r0,loop
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that there are no delay slots, and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

4.11.1 Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the fi rst instruction of that iteration up to (but not including) the cycle in which we can fetch the fi rst instruction of the next iteration. Show all instructions that are in the pipeline during these cycles (not just those from the third iteration).

Answer

and r1,r1,r2	WB								
lw r1,0(r1)	MEM	WB							
lw r1,0(r1)	EX	MEM	WB						
beq r1,r0,loop	ID	EX	MEM	WB					
lw r1,0(r1)	IF	ID	EX	MEM	WB				
and r1,r1,r2		IF	ID	EX	MEM	WB			
lw r1,0(r1)			IF	ID	EX	MEM	WB		
lw r1,0(r1)				IF	ID	EX	MEM	WB	
beq r1,r0,loop					IF	ID	EX	MEM	WB

4.11.2 How often (as a percentage of all cycles) do we have a cycle in which all five pipeline stages are doing useful work?

Answer

and r1,r1,r2	WB				
lw r1,0(r1)	MEM	WB			
lw r1,0(r1)	EX	MEM	WB		
beq r1,r0,loop	ID	EX	MEM	**WB**	
lw r1,0(r1)	IF	ID	EX	MEM	WB
and r1,r1,r2		IF	ID	EX	**MEM**
lw r1,0(r1)			IF	ID	EX
lw r1,0(r1)				IF	ID
beq r1,r0,loop					IF

위 다이어그램에서 아무것도 하지 않는 부분은

beq r1,r0,loop의 WB부분.

and r1,r1,r2의 MEM 부분입니다.

총 5 cycles에 3 cycle useful 하니 60%입니다.

4.13 This exercise is intended to help you understand the relationship between forwarding, hazard detection, and ISA design. Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath:

```
Add     r5, r2, r1
Lw      r3, 4(r5)
Lw      r2, 0(r2)
Or      r3, r5, r3
Sw      r3, 0(r5)
```

4.13.1 If there is no forwarding or hazard detection, insert nops to ensure correct execution.

Answer

Add 의 결과물이 r5에 들어가기 전 까지 Lw는 기다려야 합니다. Lw 에서 r3에 값이 로딩되기 전 까지 Or는 기다려야 합니다. 역시 SW의 r3를 사용하기 전에 OR에서 r3에 WB이 진행되어야 합니다.

Add r5, r2, r1	IF	ID	EX	MEM	WB									

Lw	r3, 4(r5)		IF	***	***	***	ID	EX	MEM	WB					
Lw	r2, 0(r2)						IF	ID	EX	MEM	WB				
Or	r3, r5, r3							IF	***	***	ID	EX	MEM	WB	
Sw	r3, 0(r5)										IF	**	**	**	ID

위 ***처럼 stall이 들어가면 되므로, 정답은

```
Add     r5, r2, r1
Nop
Nop
Nop
Lw      r3, 4(r5)
Lw      r2, 0(r2)
Nop
Nop
Or      r3, r5, r3
Nop
Nop
Nop
Sw      r3, 0(r5)
```

4.13.2 Repeat 4.13.1 but now use nops only when a hazard cannot be avoided by changing or rearranging these instructions. You can assume register R7 can be used to hold temporary values in your modified code.

Answer

Instruction의 register parameter들이 계속 연속적으로 사용되기 때문에(특히 r3), 딱히 rearrange를 할 수 없는 상황입니다. 게다가 temporary register가 주어진다고 해도 저장해놓을 만한 value도 없는 상황입니다.

4.13.3 If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when this code executes?

Answer

Add	r5, r2, r1	IF	ID	EX	MEM	WB			
Lw	r3, 4(r5)		IF	ID	EX	MEM	WB		
Lw	r2, 0(r2)			IF	ID	EX	MEM	WB	
Or	r3, r5, r3				IF	ID	EX	MEM	WB
Sw	r3, 0(r5)					IF	ID	EX	MEM	WB

두번째 lw의 EX는 forwading이 잘 이루어져서 정상적으로 처리 됩니다.

역시 네번째 OR의 EX도 두번째 lw의 MEM 결과가 forwarding되면서 정상처리 됩니다.

맨 마지막 SW에서도 OR의 EX 결과가 forwading되어서 SW의 MEM에 도달할 수 있도록 한다면 모든게 정상적으로 처리가 됩니다.

4.13.4 If there is forwarding, for the first five cycles during the execution of this code, specify which signals are asserted in each cycle by hazard detection and forwarding units in Figure 4.60.

Answer

첫 5사이클 동안의 다이어그램은 아래와 같다.

Add	r5, r2, r1	IF	ID	EX	MEM	WB
Lw	r3, 4(r5)		IF	ID	EX	MEM
Lw	r2, 0(r2)			IF	ID	EX
Or	r3, r5, r3				IF	ID
Sw	r3, 0(r5)					IF

시그널 종류를 정확하게 몰라서, 기능대로 서술하면

3번째 cycle에서 ID/EX의 rt랑 EX/MEM의 rd가 같기 때문에 forward가 이루어집니다.

4.13.5 If there is no forwarding, what new inputs and output signals do we need for the hazard detection unit in Figure 4.60? Using this instruction sequence as an example, explain why each signal is needed.

잘 모르겠습니다.