1. Instruction size: 3 bytes, byte addressable, Integer: 8bit two's complement representation

R1 = 70, R2= 80, M[70] = 70, M[71] = 71, M[72] = 72

```
300:    MOV   R2, #70      ; R2 <= 70
303:    SUB   R2, 1@R1     ; R2 <= R2 - M[R1+1]
306:    JP    $-9          ; branch if positive (branch to -9@PC)
```

1) After executing MOV and SUB instructions, what value is stored in register R2?

① R2 = R2 − M[R1 + 1] = 70 − 71 = −1

2) Show the values of flags (C, Z, S, OV) after SUB instruction.

② 70 = 01000110, −71 = 10111001

70+(−71) = −1 = 11111111

> C : 0, Z : 0, S : 1, OV : 0

3) What is the address of the instruction executed after JP instruction?

연산의 결과가 양수이면 JP명령어를 수행하라고 되어있기 때문에 연산결과가 -1이므로 JP명령어를 실행하지 않고 다음 명령어로 넘어가기 때문에 JP 다음에 수행할 명령어의 주소는 309로 예상된다.

4) Considering above program, design instruction format with 14 different operations and 13 registers. Determine instruction format, size of OP-code, number and size of operands, etc.

14가지의 다른 연산을 수행하기 위해서는 2^4=16 >= 14, opcode에 4bit를 할당해야한다.

13개의 레지스터가 있으므로 레지스터를 식별하기 위해선 2^4=16 >= 13, 4bit가 필요하다.

코드를 봤을 때 immediate, register indirect와 displacement addressing mode를 사용한다. 때문에 주소 지정 방식을 위해서는 2^2=4 >= 3, 2bit가 필요하다.

따라서 opcode+mode+operand+operand+displacement = 4+2+4+4+displacement=24이고 주소값이 들어갈 displacement 필드에는 큰 주소값들이 들어갈 경우를 생각해 최대한 많은 비트를 할당해 주는 것이 좋기 때문에 나머지 10bits를 모두 할당해준다.

opcode : 4bits / mode : 2bits / operand : 4bits / operand : 4bits / displacement : 10bits

2. A processor does not provide indirect addressing. Assume that the address of an operand is in main memory. How would you access the operand?

레지스터 간접 주소 지정 방식(Register Indirect Addressing)과 변위 주소 지정 방식(Displacement Addressing)을 활용한다. 간접 주소 지정 방식(Indirect Addressing)은 직접 주소 지정 방식(Direct Addressing)에서 주소값을 저장해야 되는 공간이 제한되는 점을 보완한다. 하지만 피연산자의 주소가 주 메모리에 있고 간접 주소 지정 방식이 제공되지 않

는 경우에는 레지스터를 활용하여 주 메모리에 접근할 수 있다. ① 레지스터 간접 주소 지정 방식(Register Indirect Addressing)에서는 명령어의 주소필드는 레지스터를 가르키고, 레지스터는 피연산자의 주 메모리의 주소값을 가르키기 때문에 주 메모리에 접근이 가능하다. ② 변위 주소 지정 방식(Displacement Addressing)에서는 명령어가 2개의 주소 공간을 갖는다. 하나는 PC를 가진 레지스터를 가르키고 또 다른 하나는 레지스터가 가지고 있는 값과 더해져 주 메모리의 주소값을 나타낸다.

3. A pipelined processor has a clock rate of 10GHz and executes a program with 10 million instructions. The pipeline has five stages and instructions are issued at a rate of one per clock.

a) What is the speedup of this processor for this program compared to a non-pipelined processor.

non-pipelined processor : $5 * 10^6 * 1/10^{10}$

pipelined processor : $(k+(n-1))\tau$ (k : 파이프라인 단계 수, n : 명령어 수)

-> $(5+(10^6-1)) * 1/10^{10}$

speed up = $(5 * 10^6 * 1/10^{10}) / ((5+(10^6-1)) * 1/10^{10}) = 5 * 10^6 / (4+10^6) \fallingdotseq 5$

b) What is the MIPS rate for the pipelined and non-pipelined processor.

$MIPS = I_c / T \times 10^6 = f / CPI \times 10^6$

non-pipelined processor : $10^6 / (5*10^6*1/10^{10}) * 10^6 = 2000$

pipelined processor : $10^6 / ((5+(10^6-1))*1/10^{10})*10^6 \fallingdotseq 9999.96$

c) List three pipeline hazards and briefly explain how these hazards affect the performance.

1) resource(structural) hazard

이미 파이프라인에 들어와 있는 두 개(혹은 그 이상)의 명령어들이 동일한 자원을 필요로 할 때 발생한다. 이에 대한 해결책은 주기억장치로 들어가는 포트를 여러 개 두거나 ALU 유니트들을 여러 개로 늘리는 것과 같이, 이용 가능한 자원을 늘리는 것과 Out of Order Execution과 같은 방법이 있다..

2) data hazard

오퍼랜드 위치에 대한 액세스에 충돌이 있을 때 발생한다. data hazard는 pipeline 사용 시에 발생하는 데 그 이유는 예를 들면 어떠한 명령어가 ADD 명령을 수행중일 때 그 다음 명령어가 ADD명령을 수행한 결과를 Fetch하려 할 때는 아직 전 명령어가 ADD명령을 수행중이기 때문에 Fetch가 불가능하다. 이러한 경우를 data hazard라 한다. 이에 대한 해결 방

법으로는 Branch Prediction과 같은 방법이 있다.

3) branch(control) hazard

파이프라인이 분기 예측에서 잘못된 결정을 하여 그 다음에 버려야 할 명령어들을 파이프라인으로 가져왔을 때 발생한다. 주로 반복문이나 조건문에서 일어난다..

4. Explain the typical distinguishing characteristics common to RISC organization.

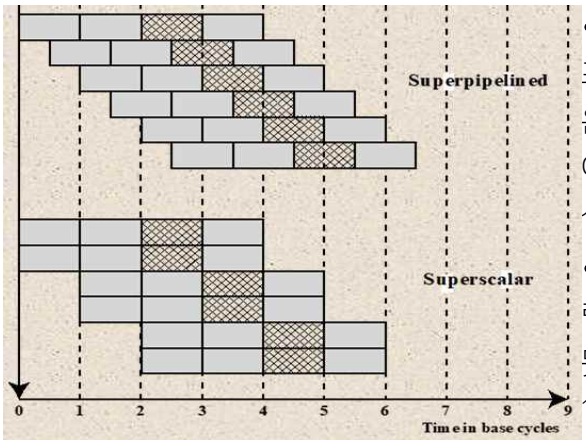
• 기계 사이클(machine cycle)당 하나의 기계어를 실행한다. RISC 기계 명령어는 CISC의 마이크로명령어보다 더 복잡하지 않아야 하고, 또한 그만큼 빨리 수행되어야 한다.

• 대부분의 연산들이 레지스터들 간에 수행되며, 기억장치를 액세스하는 동작들은 간단한 LOAD와 STORE 명령어만으로 이루어져야 한다는 것이다. 이 설계의 특징은 명령어 세트를 간략화시키며, 결과적으로 제어 유니트도 간단해진다.

• 간단한 주소지정 방식(simple addressing mode)들을 사용한다. 거의 대부분의 명령어들은 단순한 레지스터 주소지정 방식들을 사용하고 있다. 이 설계 방식은 명령어 세트와 제어 유니트를 단순화 시킨다.

• 간단한 명령어 형식(simple instruction format)들을 사용한다. 명령어 길이, 필드의 위치, 특히 연산 코드는 고정된다. 형식이 간단해져 제어 유니트가 단순해지고 명령어 인출이 최적화 된다.

5. What is the difference between the superscalar and super-pipelined approaches?



• superpipeline : 슈퍼스칼라 구조는 각 파이프라인 단계를 복제하여 여러 개를 둠으로써, 같은 파이프라인 단계에서 두 개 이상의 명령어들이 동시에 처리될 수 있다. simple pipeline보다 성능이 배가 되지만 hazard가 많이 발생한다.

• superscalar : 서로 다른 pipeline에서 독립적으로, 동시에 instruction을 수행하는 것을 뜻한다. superscalar의 궁극적인 목적은 CPI를 1보다 작게, IPC를 1보다 크게 하는 것이다.

6. Explain the purpose and contents of the interrupt vector table.

x86에서 인터럽트 처리는 인터럽트 벡터 테이블(Interrupt Vector Table)을 사용한다. 모든 형태의 인터럽트에는 어떤 번호가 지정되며, 그 번호는 인터럽트 벡터 테이블 내에서의 인덱스로 사용된다. 이 테이블은 256개의 32-비트 인터럽트 벡터들을 가지고 있는데, 이들은 그 인터럽트 수에 대한 인터럽트 서비스 루틴의 주소(segment, offset)이다.

7. Explain the difference of the instruction format between x86 processors and ARM processors

ARM은 RISC Architecture이며 x86은 CISC Architecture이다. ARM은 Load와 Store은 제외하고는 메모리를 직접 사용하며 작업을 수행할 수 없는 반면 x86은 메모리에서 직접 더 많은 작업을 수행할 수 있다.

8. Show the representation of the following data

1) character '2', 2) integer 2, and 3) real number(single precision) 2.0.

1) character '2'

char : 1byte

2) integer 2

int : 4byte

3) real number(single precision) 2.0

float : 4byte