컴퓨터구조 기말고사 (2018년도 2학기)

학번: 이름:

시험시간: 12월 18일(화) 18:00-20:00

- **답이 도출되는 과정을 명확하고 알아보기 쉽게 적을 것, 답만 쓰는 경우나 풀이 과정을 알아보기 어렵게 작성한 경우 점수 없음**

---

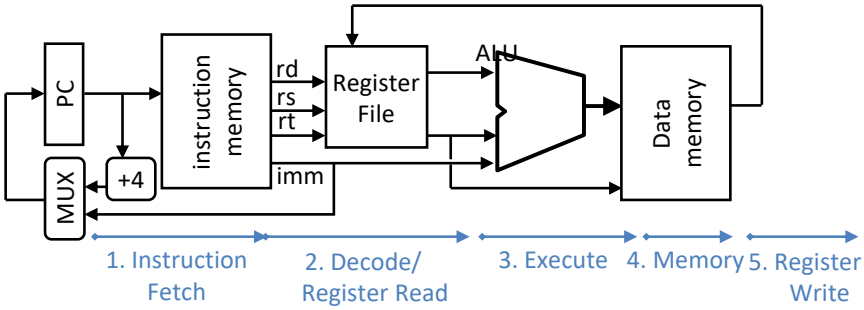
1. (30점) 아래의 문항들에 대해 해당하는 답들을 고르시오. 풀이를 쓸 필요 없이 답만 표시하면 됨.
   [주의사항] 맞은 문항은 3점, 틀린 문항은 -1점, 답을 표시하지 않은 문항들은 0점을 부여함. 그러므로 추측으로 답을 표시하지 않기를 권장함.
   a) 32-bit 주소 공간, 8 word 크기의 블록들로 구성된 32 KiB 4-way set associative cache를 사용하는 경우 tag, index, offset은 각각 어떻게 되는가?
      i. 21, 8, 3
      ii. 19, 8, 5
      iii. 19, 10, 3
      iv. 17, 10, 5

   b) 200 ps의 clock 주기, 50 clock cycle의 miss penalty, 0.02 misses/instruction의 miss rate, 1 clock cycle의 hit time을 가지고 있는 cache에서 아래의 변경 사항들 중 average memory access time (AMAT)를 가장 많이 개선하는 것은?
      i. Clock 주기를 190 ps 로 단축
      ii. Miss penalty를 40 clock cycle로 단축
      iii. Miss rate를 0.015 misses/instruction으로 단축

   c) 4-way set associative cache, random replacement, 1 KiB cache size, 16 B block size, write-back, 16-bit 주소 공간으로 구성된 cache를 구현하기 위해 필요한 총 bit 수는?
      i. $2^6 \times (2^7 + 2^3 + 2^1) = 8.625\ Kib$
      ii. $2^4 \times (2^7 + 2^3 + 2^0) = 2.140625\ Kib$
      iii. $2^4 \times (2^7 + 2^3 + 2^1) = 2.15625\ Kib$
      iv. $2^4 \times (2^7 + 6 + 2^1) = 2.125\ Kib$

   d) 2단계로 구성된 cache가 1 clock cycle의 level-1 hit time, 2% level-1 miss rate, 5 clock cycle level-2 hit time, 5% level-2 miss rate, 100 clock cycle main memory hit time을 가지고 있다. Level-1 cache는 unified cache라고 가정하자. Level-2 cache를 사용하지 않았을 때와 사용했을 때의 AMAT는 clock cycle 단위로 각각 어떻게 되는가?
      i. 3.0, 1.2
      ii. 3.1, 1.0
      iii. 2.9, 1.1
      iv. 3.0, 1.0

e) 아래와 같이 5단계로 구성된 MIP 프로세서에서 가장 적은 단계를 이용하여 수행되는 MIPS 명령어는?



i. lw
ii. jal
iii. beq
iv. addu

f) 수업시간에 배웠던 6개의 MIPS 명령어들을 수행하기 위한 프로세서는 아래와 같이 5개의 수행 단계들은 아래와 같다.

Instr Fetch	Reg Read	ALU Op	Mem Access	Reg Write
200ps	100 ps	200ps	200ps	100 ps

여기에서 load와 store 명령어들을 제외하면 'Mem Access' 단계를 제외한 4단계들로 명령어를 실행하는 경우를 파이프라인으로 구현한 후의 성능 변화를 생각해 보자. 아래의 두 문장들의 참, 거짓 여부를 판단하시오.

- ✓ Latency는 1.25배 나빠진다.
- ✓ Throughput은 3배 좋아진다.

i. 거짓, 거짓
ii. 거짓, 참
iii. 참, 거짓
iv. 참, 참

2. (10점) 아래와 같이 행렬 연산을 위한 C 코드가 주어져 있다. 각 행렬에서 같은 행에 저장되는 element들은 메모리에 연속된 주소들에 위치한다. **A와 B는 8000x8000 크기의 2차원 배열**이라고 가정하자.

```
for (i=0; i<8000; i++)
   for (j=0; j<8; j++)
      A[i][j] = B[j][0] + A[j][i];
```

(a) Temporal locality를 나타내는 변수와 그 이유를 간단히 쓰시오.

(b) Spatial locality를 나타내는 변수와 그 이유를 간단히 쓰시오.

3. (15점) 파이프라인 시스템의 각 stage들에서의 수행시간이 다음과 같이 주어져 있다.

Stage 1 = 40 ns, Stage 2 = 10 ns, Stage 3 = 30 ns, Stage 4 = 5 ns.

(a) 파이프라인 시스템의 clock frequency를 MHz 단위로 계산하시오.

(b) 파이프라인 기법을 사용하지 않고 하나의 명령어를 수행하는데 걸리는 시간은 몇 ns인가?

(c) 100개의 명령어를 수행했을 때 파이프라인 기법으로 throughput(단위 시간당 명령어 처리 개수)이 얼마나 향상되는가? (소수점 아래 둘째자리까지 계산)

4. (15 점) 5 단계 MIPS pipeline 이 주어져 있다. 이 파이프라인은 forwarding (또는 bypass) path 가 없으며 branch delay slot 도 이용하지 않는다. 그리고 branch 명령어에서 branch 의 실제 실행 여부는 execution stage 에서 결정된다고 가정한다. 다음의 코드에 대해 아래의 질문들에 답하고 이유도 간단하게 기술하시오 (답만 쓴 경우 부분 점수 없음)

```
add r0, r1, r2
sub r3, r0, r4
```

(a) 위의 명령어들을 수행하기 위해 몇 사이클의 stall 이 발생하는가?

(b) Forwarding path 가 write back stage 의 시작 시점에서 execution stage 시작 시점으로 연결되었다고 가정해보자. 몇 사이클의 stall 이 발생하는가?

(c) 모든 forwarding path 가 구현되어 있다고 가정하면 stall 사이클은 얼마가 되는가?

5. (20 점) 다음의 코드에 대해 아래의 질문들에 답하시오. 4 번 문제와 동일한 프로세서 구조를 사용하고 `r0` 는 읽기/쓰기가 가능한 레지스터라고 가정한다.

```
add r3, r4, r5
beq r0, r1, label
sub r0, r1, r2
add r4, r4, r4
```

(a) `r0 != r1` 일 때, 문제의 앞부분에서 설명했던 프로세서 구현에서는 몇 개의 명령어들이 낭비, 즉 몇 사이클의 stall 이 발생하는가?

(b) `r0 == r1` 일 때, 문제의 앞부분에서 설명했던 프로세서 구현에서는 몇 개의 명령어들이 낭비되는가?

(c) Branch 여부가 decode stage 에서 결정된다고 가정해 보자. `r0 == r1` 일 때, 몇 개의 명령어들이 낭비되는가?

(d) 프로세서에 branch delay slot 을 구현되어 있고 (즉 branch 다음 사이클에서 branch 에 독립적인 다른 명령어를 수행) branch 수행 여부는 decode stage 에서 결정된다고 가정하자. 아래의 코드에서 `r0 == r1` 일 때 파이프라인에서 프로세서 사이클의 낭비를 최소화하도록 명령어들의 실행 순서를 재배치하시오. 명령어들을 수행하기 위해 몇 사이클이 필요한가?

6. (10점) Cache block에 들어 있는 data들이 아래와 같다고 가정해 보자. CPU는 바이트 단위로 메모리를 읽는다. 그러므로 CPU의 word 크기는 한 바이트가 된다. CPU의 주소는 13bit로 이루어져 있다. 아래의 cache는 4-way set associative이고, cache block의 크기는 4-byte이며 전체 block의 개수는 32개이다.

   아래의 table에서 **모든 숫자들은 16진수이다**. "Index" 컬럼은 4개의 block으로 구성된 set들의 index를 나타낸다. "Tag" 컬럼은 각 block들의 tag 값들을 나타낸다. "V" 컬럼은 각 block들의 valid bit를 나낸다. "Byte 0-3" 컬럼은 각 block에 들어 있는 데이터들을 나타낸다. 한 block에 들어 있는 4개의 byte들은 왼쪽부터 오른쪽으로 byte 0부터 byte 3에 해당한다.

4-way Set Associative Cache																						
Index	Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3				Tag	V	Bytes 0-3			
0	84	1	ED	32	0A	A2	9E	0	BF	80	1D	FC	10	0	EF	9	86	2A	E8	0	25	44	6F	1A
1	18	1	03	3E	CD	38	E4	0	16	7B	ED	5A	02	0	8E	4C	DF	18	E4	1	FB	B7	12	02
2	84	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D8	93	1	C7	B7	AF	C2
4	84	1	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	BC	91	D5	92	28	1	80	BA	9B	F6	6B	0	48	16	81	0A
6	8B	1	5D	4D	F7	DA	29	1	69	C2	8C	74	B5	1	A8	CE	7F	DA	BF	0	FA	93	EB	48
7	84	1	04	2A	32	6A	96	0	B1	86	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

(a) 위의 cache를 사용하기 위해 13-bit 주소는 tag, index, offset의 3 부분으로 나눌 수 있다. 각각은 몇 bit가 되는가?

(b) 위의 cache를 이용하여 CPU가 주소 0x0AEE 를 접근할 때, 아래의 table에 해당하는 내용들을 **Y/N** 또는 **16진수**로 채우시오. 그리고 cache hit가 발생한다면 CPU가 읽게 될 data의 값을 "Cache Byte returned" 항목에 쓰시오, cache miss일 경우에는 "-"로 표시한다.

parameter	value
Cache Offset (CO)	
Cache Index (CI)	
Cache Tag (CT)	
Cache Hit? (Y/N)	
Cache Byte returned	