컴퓨터구조 중간고사 (2017년도 2학기)

학번: 이름:

시험시간: 12월 15일(목) 오전 9:00-11:00

아래를 반드시 읽고 나서 문제를 푸시오.

- 시험지 앞, 뒷면에 6 문제가 모두 제대로 인쇄되었는지 확인할 것
- **답이 도출되는 과정을 명확하고 알아보기 쉽게 적을 것, 답만 쓰는 경우나 풀이 과정을 알아보기 어렵게 작성한 경우 점수 없음**
- 문제들마다 난이도가 다르므로 먼저 전체를 한번 살펴본 후 시간을 잘 조절하여 문제를 풀 것

---

1. Number Representation
   (a) 0, 1, -1을 unsigned와 2의 보수 8-bit로 나타내시오. 변환할 수 없는 경우에는 "나타낼 수 없음"으로 명시할 것.

   (b) 17, -17을 unsigned와 2의 보수 8-bit로 나타내시오. 변환할 수 없는 경우에는 "나타낼 수 없음"으로 명시할 것.

2. Number Representation
   32진수를 사용한다고 가정해보자. 32개의 숫자가 필요하므로 0~9까지의 10개 외에, 알파벳 22개, A (10), B (11), C(12), D(13), E (14), F (15), G (16), H (17), I (18), J (19), K (20), L (21), M (22), N (23), O (24), P (25), Q (26), R (27), S (28), T (29), U (30), V (31)를 추가로 이용한다. 각 알파벳 뒤의 숫자는 해당하는 10진수를 나타낸다.
   위의 32진법으로 나타낸 FUN_{32}를 2진수, 16진수, 10진수로 변환하시오.

3. C Memory Management

   아래 C 코드의 문제점이 무엇인지 한두문장으로 간략하게 설명하시오.

```
int* pi = malloc(314 * sizeof(int));
if(!raspberry) pi = malloc(1 * sizeof(int));
return pi;
```

4. MIPS에서의 배열

   배열 `int arr[6] = {3, 1, 4, 1, 5, 9}`가 주어져 있고, 이 배열은 주소 `0xBFFFFF00`에서 시작한다고 가정하자. 레지스터 `$S0`는 `arr`의 주소 `0xBFFFFF00`를 저장하고 있다. 정수와 포인터 변수들은 4-byte 크기라고 가정한다. 아래의 어셈블리 코드들은 어떤 동작을 하는지 한두 문장으로 간략하게 설명하시오.

(a)

```
lw  t0, 0(s0)    # Loads arr[0] into register t0
lw  t1, 8(s0)    # Loads arr[2] into register t1
add t2, t0, t1   # Sets t2 equal to t0 plus t1
sw  t2, 4(s0)    # Sets arr[1] equal to value in t2
```

(b)

```
       add  t0, x0, x0     # Sets register t0 to 0
loop:  slti t1, t0, 6      # Sets t1 to 1 if t0 < 6, 0 otherwise
       beq  t1, x0, end    # Branches to the end if t1 is 1 (t0 >= 6)
       slli t2, t0, 2      # Sets t2 to t0 * 4 (4 is number of bytes in an integer)
       add  t3, s0, t2     # Sets t3 to the address of arr[t0] (added t2 bytes to arr)
       lw   t4, 0(t3)      # Load arr[t0] into register t4
       sub  t4, x0, t4     # Sets t4 to its negative
       sw   t4, 0(t3)      # Stores this updated value back at arr[t0]
       addi t0, t0, 1      # Increments t0 to move to the next element
       jal  x0, loop       # Jump back to the loop label
end:
```

5. Instruction Set Architecture

MIPS 명령어는 8-bit 데이터, 즉 바이트 단위의 데이터들을 다룬다. 새로운 명령어 집합(ISA)을 설계한다고 가정하자. 이번에는 8-bit 대신 4-bit 데이터, 즉 nibble 단위 데이터들을 다루는 명령어 집합을 만들려고 한다. 새로운 ISA에서 워드의 크기는 6-nibble, 즉 24-bit라고 하자. 그리고 레지스터들도 6-nibble 크기이며, 명령어들도 6-nibble 크기로 구성된다. 메모리는 nibble 주소 단위로 접근한다. 새로운 nibble 주소 체계에서 명령어들과 데이터 워드들의 시작 주소는 6의 배수로 정렬되어 있다. 즉 워드 0은 nibble 0, 워드 1은 nibble 6, 워드 2는 nibble 12 등의 주소에서 시작하게 된다.

32-bit MIPS ISA의 명령어 형식과 필드들을 새로운 ISA를 설계하는데 적용하려고 한다. 새로운 명령어들와 워드 데이터들이 24-bit 크기이고 메모리 주소가 nibble 단위로 관리되는 것을 빼면 많은 부분들이 유사하다.

(a) 아래의 그림에 새로운 24-bit ISA의 R-type와 I-type의 명령어 형식의 필드들의 크기를 bit 단위로 채우시오.

opcode	rs	rt	rd	shamt	funct
6					

opcode	rs	rt	imm
6			

(b) rs, rt, rd 필드의 크기를 고려할때 새로운 ISA에서 사용할 수 있는 레지스터의 갯수는?

(c) 새로운 ISA에서 주어진 opcode에서 사용 가능한 function의 최대 갯수는?

(d) PC에 **nibble 주소**로 1566 가 저장되어 있다면, branch 명령어를 통해 점프할 수 있는 가장 큰 nibble 주소는 십진수로 얼마인가?

(e) 새로운 ISA 형식으로 인코딩된 아래의 24-bit 명령어를 MIPS 명령어로 변환하시오.

0x8C2408

변환시 레지스터 표기는 $0, $1,... 등과 같은 레지스터 번호를 사용하고 MIPS ISA와 같은 opcode를 사용한다고 가정하시오. (opcode는 다음을 참고: lw 35, addiu 4, j 8)