The problem set is divided into two parts. The first part includes problems 1,2 and 3, and the second part includes problems 4, 5, and 6.

1. (15%) Let S be a set of n elements. The *union-find* problem on the set S consists of a sequence of `union(x,y)` and `find(z)` instructions. Initially, each element in S is a set by itself. The instruction `union(x,y)` makes the set containing x and the set containing y into one set. The `find(z)` reports the name of the set in which z belongs. The name of a set can be any elements in that set, but it must be consistent. That is, any element in the same set should get the same name, and the name cannot be changed, except it is unioned to another set. Design data structures and algorithms for the problem so that both instructions can be executed efficiently for large $|S|$. Analize the time complexity of `union(x,y)` and `find(z)`.

2. (20%) Euclid's algorithm is an efficient algorithm for finding the greatest common divisor of two positive integers. It can be described in a C-like procedure as follows.

   $$\text{input}(a, b); \text{ while } (b \neq 0)\ \{c = a\%b;\ a = b;\ b = c;\};\ \text{output}(a);$$

   Assume that $a > b$. Show that the algorithm will always terminate in $O(\log a)$ iterations.
   Hint: Show that at the end of the r-th iteration, the value of a is at most $\left(\frac{1+\sqrt{5}}{2}\right)^{k-r}$, where k is the total number of iterations.

3. (15%) Let $G = (V, E)$ be a connected graph. Let $w : E \rightarrow R^+$ be a nonegative weight function defined on the edges of G. Let H be a subgraph of G. The weight of H is defined to be the sum of the weights on the edges of H. A minimum spanning tree of G is a spanning tree with mimimum weight. The Prim's algorithm and the Kuskal's algorithm are the most commonly used algorithms to find a minimum spanning tree for G. Described the two algorithms and design a data structure for each of these two algorithms so that each of the algorithm can be run efficiently.

4. (20%) The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced. The following briefly explains the different addressing modes:

(a) *immediate mode*: the operand is specified in the instruction itself.

(b) *register mode*: the address field specifies a processor register.

(c) *register-indirect mode*: the instruction specifies a processor register whose content gives the address of the operand in memory.

(d) *direct addressing mode*: the address field of the instruction gives the address of the operand in memory.

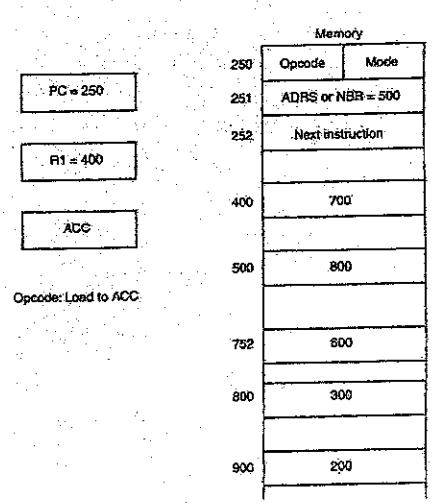
(e) *indirect addressing mode*: the address field of the instruction gives the address at which the effective address is stored in memory.

(f) *indexed addressing mode*: the content of an index register is added to the address part of the instruction to obtain the effective address.

(g) *relative addressing mode*: the address part of the instruction is added to the content of PC (program counter) to obtain the effective address.

In the following figure, the instruction in addresses 250 and 251 is "load to ACC" with the address field ADRS (or an operand NBR) equal to 500. The mode filed specifies the addressing mode. The program counter (PC) has the number 250 for fetching this instruction and its content is incremented to 252 after fetching the instruction. The processor register R1 (with content of 400) is assumed to be used in register mode, register-indirect mode and indexed addressing mode. The ACC (accumulator) receives the results after the instruction is executed. Please complete the following table by filling in the effective address and the contents of the ACC after executing the instruction in different addressing modes.

Addressing mode	Effective address	Contents of ACC
direct		
immediate		
indirect		
relative		
index		
register	NA	
Register indirect		



5. Make comparison for each of the following terminologies:

5.1 (5%) external interrupt vs. internal interrupt vs. software interrupt

5.2 (5%) direct-map cache vs. set-associate cache

6. Assume that there are four pipeline stages: fetch, decode, execute and write-back in a microprocessor.

6.1 (5%) What is the problem when executing the following instruction sequence?

```
(1) MOV   R1, R5   /* move the content of register R5 to register R1 */
(2) ADD   R2, R1, R6 /* add the contents of R1 and R6 and store the result in R2 */
(3) ADD   R3, R1, R2
```

6.2 (5%) Give two different methods (one in software and the other in hardware) to solve the above problem and explain how they work by drawing the corresponding pipelined sequence of instruction with the horizontal axis showing the time (in clock cycles) and the vertical axis showing the program execution order (in instructions).

6.3 (5%) What is the problem when executing the following instruction sequence?

```
(1) BZ    R1, 18     /* if R1=0, a branch to the instruction in location 20
                        (PC+18=2+18=20) */
(2) MOV   R2, R3
(3) MOV   R1, R2
(4) MOV   R4, R2
                ....
(20) MOV  R5, R6
```

6.4 (5%) Give two methods (one in software and the other in hardware) to solve the above problem and explain how they work.