(a) [1.4 pt] Consider a processor with a 4-way set-associative cache with one-word blocks and a total cache size of 32 words. The cache uses a least recently used replacement policy and is initially empty.

The following sequence of decimal word address references is seen by the cache.

30, 86, 53, 61, 29, 37, 30, 45, 6, 22, 14, 6, 53, 29, 78, 22, 70, 61, 54, 78, 45, 30, 61, 37, 45, 6, 29

i. Indicate whether each address reference is a hit or a miss.

Address	Set	Hit/Miss
30		
86		
53		
61		
29		
37		
30		
45		
6		
22		
14		
6		
53		
29		
78		
22		
70		
61		
54		
78		
45		
30		
61		
37		
45		
6		
29		

ii. Show the final cache contents.

(b) [2 pt] Consider the following short program executing on a simple 5-stage pipeline (Fetch, Decode, Execute, Memory, Writeback). In the notation used below, $N denotes register N. For arithmetic instructions, the destination register is listed first, followed by the source operands. For example, add $3, $2, $1 adds the contents of registers 1 and 2 and stores the result in register 3.

```
(1) lw   $1, 40($6)
(2) add $6, $2, $2
(3) sw   $6, 50($1)
(4) add $4, $5, $6
(5) lw    $6, 10($4)
```

i. Identify all the data dependencies in the code given above.

ii. Assume a pipeline that does not implement forwarding. Identify which dependencies from part (i) will cause data hazards if NOPs are not inserted.

iii. If NOP instructions are inserted to avoid hazards in absence of forwarding hardware, how many cycles does it take to execute the code?

iv. Now assume that forwarding paths are added to the pipeline and NOPs are added only in cases where forwarding does not resolve a hazard. With forwarding, how many cycles will it take to execute the code?

(c) [0.6 pt] A student runs a serial (non-parallel) program on a single core of a 128-core processor. The student uses gprof to profile the code and observes the following output. (gprof shows the percentage of execution time spent executing each function in the program.)

NAME	TIME	%
work	4163	98
play	85	2

i. Observing that most of the execution time is spent executing the "work" function, the student decides to write a new version of the program in which the "work" function is replaced by a parallel implementation of the function. What is the fastest execution time the student can expect when running the parallel version of the program on the 128-core processor?

ii. Given a processor with unlimited cores, what is the maximum speedup that can be achieved by the student's code?