1. (Short answer: 2 points each—10 points total) Fill in the word or phrase that best matches the description provided. In most cases, what is needed is a general answer, not a specific instance of the concept.

(a) ________________  Grabbing a value from somewhere in the pipeline after its computed but before its written back.

(b) ________________  Virtual nop inserted into the pipeline.

(c) ________________  Describes pattern in which C arrays are stored in memory.

(d) ________________  Forwarding won’t solve this data hazard because the value needed is read from memory.

(e) ________________  Y86 statement type for which the pipeline does not try to predict the next PC value.

2. _____ (2 points) All of the following are differences between the sequential and pipelined Y86 implementations except:

(a) the pipelined version has more sequential logic;
(b) the clock speed of the sequential version is probably slower;
(c) the PC after an instruction executes may differ;
(d) there are no control hazards in the sequential version;
(e) all of these are differences.
3. (8 points) Match each C function on the left with the assembly code on the right that implements it. Write A-D on the line.

alpha: assem ________    gamma: assem ________
beta: assem ________      delta: assem ________

```c
struct node {
    long x;
    char str[16];
    struct node *next;
};

long alpha(struct node *ptr) {
    return ptr->x;
}

cchar *beta(struct node *ptr) {
    ptr = ptr->next;
    return ptr->str;
}

cchar gamma(struct node *ptr) {
    return ptr->str[7];
}

cchar *delta(struct node *ptr) {
    return &ptr->str[7];
}
```

```assembly
assemA:
    movq 24(%rdi), %rax
    addq $8, %rax
    ret

assemB:
    leaq 15(%rdi), %rax
    ret

assemC:
    movzbl 15(%rdi), %eax
    ret

assemD:
    movq (%rdi), %rax
    ret
```

Page total: ____/8