Pre-Lecture 9

Due Sep 18 at 9am **Points** 9 **Questions** 5 **Available** until Sep 18 at 9am
Time Limit None **Allowed Attempts** 2

Instructions

Take this quiz ***after you have watched the required videos and/or read the associated sections of the textbook***. See **Lecture 9: Optimization II**.

You may attempt this quiz twice. Incorrect responses are marked after each attempt. Correct answers are revealed at the start of class for this lecture.

Carefully note the deadline for responses. Submissions are not accepted after the deadline, and there is no grace period.

This quiz was locked Sep 18 at 9am.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	2,301 minutes	8 out of 9

Score for this attempt: **8** out of 9
Submitted Sep 18 at 7:09am
This attempt took 2,301 minutes.

Question 1 — 3 / 3 pts

Consider the following C procedure:

```
int array_product(int a[], int length) {
  int i, product;

  for(i = 0, product = 1; i < length; i++)
    product = product * a[i];

  return product;
}
```

Fill in the blanks below to complete a version of the procedure with a 2x2 version of the loop (i.e., unrolled by a factor of two with two accumulators).

```
int array_product(int a[], int length) {
  int i, product0, product1;

  for(i = 0, product0 = 1, product1 = 1; i < length-1; i += ___) {
    product0 = ___ * a[i];
    product1 = ___ * a[___];
  }

  for(; i < length; i += ___)
    product0 = ___ * a[___];

  return product0 * product1;
}
```

Blank 1: 2

Blank 2: product0

Blank 3: product1

Blank 4: i+1

Blank 5: 1

Blank 6: product0

Blank 7: i

(NOTES: Blanks are read left to right and top to bottom. ***Exact*** C code is expected.)

Answer 1:

Correct!

2

Answer 2:

Correct!

product0

Answer 3:

Correct!

product1

Answer 4:

Correct!

i+1

orrect Answer i + 1

Answer 5:

Correct! 1

Answer 6:

Correct! product0

Answer 7:

Correct! i

Question 2

1 / 1 pts

Consider again the C procedure from Question 1.

Assume a machine for which the latency of integer addition is 1.0 cycle with issue time of 0.33 cycles, and the latency of integer multiplication is 3.0 cycles with issue time of 1.0 cycle.

The lower bound on the CPE for the original procedure is 3.0. What is the lower bound on the CPE for the new procedure with a 2x2 version of the loop?

Correct! 1.5

orrect Answers 1.5 (with margin: 0)

Question 3

0 / 1 pts

Consider again Questions 1 and 2.

For an NxN version of the loop (unrolled by a factor of N with N accumulators), we do not expect to see further improvement on the performance of the C procedure array_product. What is the value of N?

In this case, you may assume that register pressure does not limit N.

'ou Answered

4

orrect Answers

3

three

Three

Question 4

1 / 1 pts

Assuming that the elements of a collection are ***floating-point*** numbers, what is the expected CPE for the following loop on a Haswell processor?

```
for (i = 0; i < collection->count; i++) {
   sum1 *= collection->elements[i];
   sum2 *= collection->elements[collection->count-i-1];
}
```

- 3 CPE
- 5 CPE

Correct!

- 10 CPE
- 11 CPE

Question 5

3 / 3 pts

Here's a function (that you may recognize as a hailstone computation):

```
long iterate(long v, int steps)
{
  while (steps--) {
    if (v & 0x1) {
      v = 3*v + 1;
```

```
    } else {
      v = v >> 1;
    }
  }

  return v;
}
```

Suppose that this function is called with 1 as the first argument and a large N as the second argument. Roughly how long should the function take in cycles on a Haswell processor?

Note: Assume that the compiler implements `3*v+1` using a single `lea` instruction that takes 3 cycles (like an integer multiplication by itself), instead of a multiplication followed by an addition.

- about N cycles
- Correct! about 1.67*N cycles
- about 3*N cycles
- about N^2 cycles

Quiz Score: **8** out of 9