HW4

● Graded

Student

Adam Fenjiro

Total Points

29.5 / 57 pts

Question 1

Q1

6 / 6 pts

✔ **– 0 pts** Correct

– 1 pt Partially correct variables under structural equivalence

– 1 pt Partially correct variables under strict name equivalence

– 1 pt Partially correct variables under loose name equivalence

– 2 pts Missing/Incorrect variables under structural equivalence

– 2 pts Missing/Incorrect variables under strict name equivalence

– 2 pts Missing/Incorrect variables under loose name equivalence

Question 2

Q2

1.5 / 5 pts

– 0 pts Correct

– 1 pt Partially correct explanation for total space

✔ **– 1.5 pts** Incorrect / missing total space consumed by the array

✔ **– 2 pts** Incorrect / missing explanation

Question 3

Q3 **4** / 26 pts

3.1 **a** **2** / 16 pts

- **– 0 pts** Correct
- **– 1 pt** Minor errors in offset calculation for column-major layout
- **– 1 pt** Minor errors in offset calculation for row-pointer layout
- **– 1 pt** Minor errors in x86 assembly code for column-major layout
- **– 1 pt** Minor errors in x86 assembly code for row-pointer layout
- **– 1 pt** Use of global offset instead of stack pointer for column-major layout
- **– 1 pt** Use of global offset instead of stack pointer for row-pointer layout
- ✔ **– 3 pts** Incorrect / missing x86 assembly for column major
- ✔ **– 3 pts** Incorrect / missing x86 assembly for row pointer
- ✔ **– 4 pts** Missing/incorrect offset address calculation for column major
- ✔ **– 4 pts** Missing/incorrect offset address calculation for row pointer
- **– 16 pts** No attempt

3.2 **b** **2** / 10 pts

- **– 0 pts** Correct
- **– 1 pt** Use of global pointer instead of stack pointer
- **– 2 pts** Partially correct memory layout for given variables
- ✔ **– 4 pts** Incorrect/missing memory layout for given variables
- **– 2 pts** Partially correct address calculation for r.A[2,j].B[k]
- ✔ **– 4 pts** Incorrect/missing address calculation for r.A[2,j].B[k]
- **– 1 pt** Partially correct x86 assembly
- **– 2 pts** Incorrect/missing x86 assembly
- **– 10 pts** No attempt

Question 4

Q4 **10** / 10 pts

✔ **– 0 pts** Correct

– 1 pt Missing output

– 1 pt acknowledges output is different

– 1 pt No attempt to answer the output change question

– 1.5 pts Partially correct explanation about discrepancy

– 2 pts incorrect / missing reason as to why output is not different

– 3 pts Missing output analysis for the first two lines

– 3 pts Missing output analysis for the last two lines

– 10 pts Incorrect

Question 5

Q5 **8** / 10 pts

– 0 pts Correct

✔ **– 2 pts** No mention of loose name equivalence in phenomenon

– 2 pts Partially correct code changes

– 4 pts Incorrect/missing code changes

– 10 pts No attempt

Questions assigned to the following page: 1 and 2

Adam Fenjiro
CS4121, HW4

Problem 1:

- **Structural equivalence:** I think that all four variables are structurally equivalent since they, all, share the same data structure being an integer array of 10 elements.

- **Strict name equivalence:** I think that a and b are strictly name equivalent, d is not strictly name equivalent with them, and c is also not strictly name equivalent. This is because a and b are both of type T, c is of type S, and since d is defined explicitly as an array of integers and does not share a named type with a, b, or c, making it not strictly name equivalent with any of them.

- **Loose name equivalence:** a, b, and c are loosely name equivalent because they are aliases of the same type. But d is not a loosely name equivalent with a, b, or c because it lacks a named type association even though it has a similar structure.

Problem 2:

- Size of short (s, t) = 2 * 2 = 4 bytes
- Size of char (c, d) = 1 * 2 = 2 bytes
- Size of real (r) = 8 bytes
- Size of integer (i) = 4 bytes

Thus total size of array = (4+ 2+ 8+ 4) * 10 = 180 bytes

Questions assigned to the following page: 3.1 and 3.2

Problem 3:

a.

```
Unset
LOAD R1, i
SUB R1, 10
ADD R1, 182
SHL R1, 3
ADD R1, Base(A)
LOADFP R2, 0(R1)
STOREFP R2, x
```

Since this is the address for A[3, i]:
Address(A[3,i]) = Base(A) + ((3 - 1) * 91 + (i - 10)) * 8

So the code I provided will calculate the offset from the base of A, then it will load the value at that address into the register to then store it into the variable x.

b.

```
Unset
LOAD R1, j
SUB R1, 10
ADD R1, 11
MUL R1, 9
ADD R1, k
ADD R1, Base(r) + 5
LOAD R2, 0(R1)
```

Since the address for r.A[2,j].B[k]:
Address(r.A[2,j].B[k]) = Base(r) + 4 + 1 + ((2 - 1) * 11 + (j - 10)) * 9 + k

So the code that I provided get the offset from the base of r, then loads the character at that address into a register.

Question assigned to the following page: 4

Problem 4:

Here is the output:

```
Unset
1
1.5
2
2.0
```

In type.c, i1 and i2 are integers while f1 and f2 are floats. This means that i1 and i2 cannot have any decimal point while f1 and f2 must have the decimal value. Here is how the output works:

- i1 => 1 since it is an integer
- i1/i2 => 3/2 => 1 and not 1.5
  This is because it's integer division
  1.0/2 => 0.5
  This is because one of the operands is a float
  Adding these two gives => f1 = 1+0.5 = 1.5
- I1 => 2 since it is an integer
- f1/i2 => 3.0/3 = 1.5
  1.0/2 => 0.5
  This is because one of the operands is a float
  Adding these two gives => f1 = 1.5 + 0.5 = 2.0

Will the output be different if we calculate 1.0/2 first in the two assignments to f1?
No I don't think the output will not be different if we calculate 1.0/2 first, this is because compiler sees division whether any of the operands is integer or float.

Question assigned to the following page: 5

Problem 5:

Here is the new updated code:

```C/C++
#include <stdio.h>
struct S1 {
int a;
int b;
} s1, s3;

struct S2 {
int b;
int a;
} s2;

typedef struct S1 S;
S s4;

main(){
s1.a = 10;
s1.b = 20;
s2.a = s1.a;
s2.b = s1.b;
s3 = s1;
s4 = s1;
printf("s2.a = %d; s2.b = %d\n", s2.a, s2.b);
printf("s3.a = %d; s3.b = %d\n", s3.a, s3.b);
printf("s4.a = %d; s4.b = %d\n", s4.a, s4.b);}
```

I noticed that the error I was getting was because the structure of s1 and s2 were different data types, making them incompatible when assigning struct s2 from struct s1. In the code above, I added theses changes which fixed the error that I spoke about:

```C/C++
s2.a = s1.a;
s2.b = s1.b;
```