COMP 235 — Assembly Study

You've been given a short C program along with the assembly generated from that file. Analyze the assembly and determine the answers to the following:

1. For *main*
   (a) attribute all lines of assembly to lines of C or to function call prologue and epilogue code done at the assembly level. Label assembly line numbers with C line numbers.
   (b) determine the size of the stack frame in bytes
   (c) determine the location within the frame for each of the 3 arrays. List the locations relative to %rbp. For example, $[-10, -5]$ would refer to bytes %rbp-10 through %rbp-5.
   (d) draw a diagram showing how each byte of the stack frame is used. If bytes are allocated on the stack but not directly used, label them as unused.
   (e) determine the primary operand specifier for the variables A, B, and C.
2. For *initAll*
   (a) attribute all lines of assembly to lines of C or to function call prologue and epilogue code done at the assembly level. Label assembly line numbers with C line numbers.
   (b) determine how much stack space is used
   (c) determine the primary operand specifier for variables a,b,c,i, and j.
3. For *matmul*
   (a) attribute all lines of assembly to line of C or to function call prologue and epilogue code done at the assembly level. Label assembly line numbers with C line numbers.
   (b) determine how much stack space is used
   (c) determine the primary operand specifier for variables a,b,c,i,j, and k

We'll spend some time in class getting started on this. This exercise is open book, open notes, open professor, open classmates, but not open internet. Do not consult with AI.

matmul.c

```c
#include <stdio.h>

void mult(int lhs[][4],int rhs[][4],int prod[][4]){
    int i,j,k;
    int sum;
    for(i=0;i<4;i++){
        for(j=0;j<4;j++){
            for(k=0;k<4 ;k++){
                prod[i][j] += lhs[i][k]*rhs[k][j];
            }
        }
    }
}

void printMat(int m[][4]){
    int i,j;
    for(i=0;i<4;i++){
        for(j=0;j<4;j++){
            printf("%3d ",m[i][j]);
        }
        printf("\n");
    }
}

void initAll(int a[][4], int b[][4], int c[][4]){
    int i,j;
    for(i = 0; i < 4; i++){
        for(j=0; j<4; j++ ){
            c[i][j] = 0;
            a[i][j] = 4*i + j;
            b[i][j] = 16-(4*i+j);
        }
    }
}

int main(){
    int A[4][4];
    int B[4][4];
    int C[4][4];

    initAll(A,B,C);

    printMat(A);
    printMat(B);
    printMat(C);

    mult(A,B,C);
    printMat(C);


```

```
51 }
```

matmul.s

```
    .file   "matmul.c"
    .text
    .globl  mult
    .type   mult, @function
mult:
    endbr64
    pushq   %rbp
    movq    %rsp, %rbp
    movq    %rdi, -24(%rbp)
    movq    %rsi, -32(%rbp)
    movq    %rdx, -40(%rbp)
    movl    $0, -12(%rbp)
    jmp .L2
.L7:
    movl    $0, -8(%rbp)
    jmp .L3
.L6:
    movl    $0, -4(%rbp)
    jmp .L4
.L5:
    movl    -12(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -40(%rbp), %rax
    addq    %rax, %rdx
    movl    -8(%rbp), %eax
    cltq
    movl    (%rdx,%rax,4), %esi
    movl    -12(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -24(%rbp), %rax
    addq    %rax, %rdx
    movl    -4(%rbp), %eax
    cltq
    movl    (%rdx,%rax,4), %edx
    movl    -4(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rcx
    movq    -32(%rbp), %rax
    addq    %rax, %rcx
    movl    -8(%rbp), %eax
    cltq
    movl    (%rcx,%rax,4), %eax
    movl    %edx, %ecx
    imull   %eax, %ecx
    movl    -12(%rbp), %eax
```

```
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -40(%rbp), %rax
    addq    %rax, %rdx
    addl    %esi, %ecx
    movl    -8(%rbp), %eax
    cltq
    movl    %ecx, (%rdx,%rax,4)
    addl    $1, -4(%rbp)
.L4:
    cmpl    $3, -4(%rbp)
    jle .L5
    addl    $1, -8(%rbp)
.L3:
    cmpl    $3, -8(%rbp)
    jle .L6
    addl    $1, -12(%rbp)
.L2:
    cmpl    $3, -12(%rbp)
    jle .L7
    nop
    nop
    popq    %rbp
    ret
    .size   mult, .-mult
    .section    .rodata
.LC0:
    .string "%3d "
    .text
    .globl  printMat
    .type   printMat, @function
printMat:
    endbr64
    pushq   %rbp
    movq    %rsp, %rbp
    subq    $32, %rsp
    movq    %rdi, -24(%rbp)
    movl    $0, -8(%rbp)
    jmp .L9
.L12:
    movl    $0, -4(%rbp)
    jmp .L10
.L11:
    movl    -8(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -24(%rbp), %rax
    addq    %rax, %rdx
    movl    -4(%rbp), %eax
    cltq
```

```
    movl    (%rdx,%rax,4), %eax
    movl    %eax, %esi
    leaq    .LC0(%rip), %rax
    movq    %rax, %rdi
    movl    $0, %eax
    call    printf@PLT
    addl    $1, -4(%rbp)
.L10:
    cmpl    $3, -4(%rbp)
    jle .L11
    movl    $10, %edi
    call    putchar@PLT
    addl    $1, -8(%rbp)
.L9:
    cmpl    $3, -8(%rbp)
    jle .L12
    nop
    nop
    leave
    ret
    .size   printMat, .-printMat
    .globl  initAll
    .type   initAll, @function
initAll:
    endbr64
    pushq   %rbp
    movq    %rsp, %rbp
    movq    %rdi, -24(%rbp)
    movq    %rsi, -32(%rbp)
    movq    %rdx, -40(%rbp)
    movl    $0, -8(%rbp)
    jmp .L14
.L17:
    movl    $0, -4(%rbp)
    jmp .L15
.L16:
    movl    -8(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -40(%rbp), %rax
    addq    %rax, %rdx
    movl    -4(%rbp), %eax
    cltq
    movl    $0, (%rdx,%rax,4)
    movl    -8(%rbp), %eax
    leal    0(,%rax,4), %ecx
    movl    -8(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -24(%rbp), %rax
```

```
    addq    %rax, %rdx
    movl    -4(%rbp), %eax
    addl    %eax, %ecx
    movl    -4(%rbp), %eax
    cltq
    movl    %ecx, (%rdx,%rax,4)
    movl    -8(%rbp), %eax
    leal    0(,%rax,4), %edx
    movl    -4(%rbp), %eax
    leal    (%rdx,%rax), %esi
    movl    -8(%rbp), %eax
    cltq
    salq    $4, %rax
    movq    %rax, %rdx
    movq    -32(%rbp), %rax
    addq    %rax, %rdx
    movl    $16, %eax
    subl    %esi, %eax
    movl    %eax, %ecx
    movl    -4(%rbp), %eax
    cltq
    movl    %ecx, (%rdx,%rax,4)
    addl    $1, -4(%rbp)
.L15:
    cmpl    $3, -4(%rbp)
    jle .L16
    addl    $1, -8(%rbp)
.L14:
    cmpl    $3, -8(%rbp)
    jle .L17
    nop
    nop
    popq    %rbp
    ret
    .size   initAll, .-initAll
    .globl  main
    .type   main, @function
main:
    endbr64
    pushq   %rbp
    movq    %rsp, %rbp
    subq    $208, %rsp
    movq    %fs:40, %rax
    movq    %rax, -8(%rbp)
    xorl    %eax, %eax
    leaq    -80(%rbp), %rdx
    leaq    -144(%rbp), %rcx
    leaq    -208(%rbp), %rax
    movq    %rcx, %rsi
    movq    %rax, %rdi
    call    initAll
    leaq    -208(%rbp), %rax
```

```
    movq    %rax, %rdi
    call    printMat
    leaq    -144(%rbp), %rax
    movq    %rax, %rdi
    call    printMat
    leaq    -80(%rbp), %rax
    movq    %rax, %rdi
    call    printMat
    leaq    -80(%rbp), %rdx
    leaq    -144(%rbp), %rcx
    leaq    -208(%rbp), %rax
    movq    %rcx, %rsi
    movq    %rax, %rdi
    call    mult
    leaq    -80(%rbp), %rax
    movq    %rax, %rdi
    call    printMat
    movl    $0, %eax
    movq    -8(%rbp), %rdx
    subq    %fs:40, %rdx
    je  .L20
    call    __stack_chk_fail@PLT
.L20:
    leave
    ret
    .size   main, .-main
    .ident  "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"
    .section    .note.GNU-stack,"",@progbits
    .section    .note.gnu.property,"a"
    .align 8
    .long   1f - 0f
    .long   4f - 1f
    .long   5
0:
    .string "GNU"
1:
    .align 8
    .long   0xc0000002
    .long   3f - 2f
2:
    .long   0x3
3:
    .align 8
4:

```