```
    .globl  copy
copy:
# A in rdi, C in rsi, N in edx
    xorl %eax, %eax         # set eax to 0
# since this function is a leaf function, no need to save caller-saved registers rcx and r8
    xorl %ecx, %ecx         # row number i is in ecx -> i = 0

# For each row
rowLoop:
    movl $0, %r8d           # column number j in r8d -> j = 0
    cmpl %edx, %ecx         # loop as long as i - N < 0
    jge doneWithRows

# For each cell of this row
colLoop:
    cmpl %edx, %r8d         # loop as long as j - N < 0
    jge doneWithCells

# Compute the address of current cell that is copied from A to C
# since this function is a leaf function, no need to save caller-saved registers r10 and r11
# Memory computation: A[i][j] = A + (i * C * L) + (j * L) = A + L * ( (i * C)  + j )
#         if C and R => N      = A + L * (i*N + j)
    movl %edx, %r10d        # r10d = N
    imull %ecx, %r10d       # r10d = i*N
    addl %r8d, %r10d        # i*N + j
    imull $1, %r10d         # r10 = L * (i*N + j) -> L is char (1Byte)
    movq %r10, %r11         # r11 = L * (i*N + j)
    addq %rdi, %r10         # r10 = A + L * (i*N + j)
    addq %rsi, %r11         # r11 = C + L * (i*N + j)

# Copy A[L * (i*N + j)] to C[L * (i*N + j)]
    movb (%r10), %r9b       # temp = A[L * (i*N + j)]
    movb %r9b, (%r11)       # C[L * (i*N + j)] = temp

    incl %r8d               # column number j++ (in r8d)
    jmp colLoop             # go to next cell

# Go to next row
doneWithCells:
    incl %ecx               # row number i++ (in ecx)
    jmp rowLoop             # Play it again, Sam!

doneWithRows:               # bye! bye!
    ret
```