The Matrix Flippy-Flop

Changes and Adjustments from Phase 2:

I utilized the generateRandomNumbers function that you wrote to expand the size of the input matrix. The matrix is now a 2^8 by 2^8 (65536 elements total) in size. I will assume that the matrix's row and column sizes are divisible by 4.

Benchmarking the original program

I am using the laptop in the engineering building hallways with no adjustment to the clock speed.

I am benchmarking the two printMatrix function calls and the flippyFlop function call, plus a few unrelated instructions that shouldn't affect the timing. On the updated, unoptimized code, I get the following results.

6,520 ms	**6,456 ms**	**6,538 ms**	**Average: 6,504.67 ms**

```
89          mov rcx, offset originalPrompt      ; 1st Parameter - Labeling the original matrix
90          sub rsp, 32                         ; Shadow Space
91          sub rsp, 8                          ; Stack is not aligned
92          call printf                         ; Print the label
93          add rsp, 40                         ; Clean up stack
94          call printMatrix                    ; Print the matrix in a nice form
95
96          call flippyFlop                     ; Flippyflop the matrix
97
98          mov rcx, offset transposePrompt     ; 1st Parameter - Labeling the transpose matrix
99          sub rsp, 32                         ; Shadow Space
100         sub rsp, 8                          ; Stack is not aligned
101         call printf                         ; Print the label
102         add rsp, 40                         ; Clean up stack
103         call printMatrix                    ; Print the matrix in a nice form
104
105         mov rcx, 0                          ; Exit code of 0
106         call ExitProcess
```

Optimization 1 – Loop Unrolling

I first unrolled the inner loop of my printMatrix function so that it would print 4 elements at a time, I then unrolled the outer loop so that it would iterate through 4 rows each time. I did not unroll my flippyFlop function's loop because the inner loop increases in the number of iterations per each outer loop iteration so I can't assume anything about the divisibility of the number of iterations each time. And the outer loop iterates n-1 times so assuming that it is even contradicts the assumption of the size's divisiblity for the printMatrix function. I got the following results when I ran my program 3 times.

6,492 ms	**6,454 ms**	**6,421 ms**	**Average: 6,455.67 ms**	**Speed Increase: ~ 1%**

```
22          ; Assumes the row size is divisible by 4
23          printOuter:
24              mov r15, 0                      ; r15 = Column counter
25              ; Assumes the column size is divisible by 4
26              printInner1:
27                  mov r13, myMatrix           ; The address of the matrix in memory
28                  ; 1st print
29                  ; printf(myString, rdx)
30                  mov rcx, offset myString    ; 1st Parameter - string
31                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
32                  sub rsp, 32                 ; Shadow Space
33                  ;sub rsp, 8                 ; If stack is not aligned
34                  call printf                 ; Print the matrix element
35                  add rsp, 32                 ; or 40; Clean up stack
36
37                  inc r12                     ; Increment array counter
38
39                  ; 2nd print
40                  ; printf(myString, rdx)
41                  mov rcx, offset myString    ; 1st Parameter - string
42                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
43                  sub rsp, 32                 ; Shadow Space
44                  ;sub rsp, 8                 ; If stack is not aligned
45                  call printf                 ; Print the matrix element
```

```
46                  add rsp, 32                 ; or 40; Clean up stack
47
48                  inc r12                     ; Increment array counter
49
50                  ; 3rd print
51                  ; printf(myString, rdx)
52                  mov rcx, offset myString    ; 1st Parameter - string
53                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
54                  sub rsp, 32                 ; Shadow Space
55                  ;sub rsp, 8                 ; If stack is not aligned
56                  call printf                 ; Print the matrix element
57                  add rsp, 32                 ; or 40; Clean up stack
58
59                  inc r12                     ; Increment array counter
60
61                  ; 4th print
62                  ; printf(myString, rdx)
63                  mov rcx, offset myString    ; 1st Parameter - string
64                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
65                  sub rsp, 32                 ; Shadow Space
66                  ;sub rsp, 8                 ; If stack is not aligned
67                  call printf                 ; Print the matrix element
68                  add rsp, 32                 ; or 40; Clean up stack
69
70                  inc r12                     ; Increment array counter
71                  add r15, 4                  ; Increment column counter
72                  cmp r15, colSize            ; Compare column counter to colSize
73                  jnz printInner1             ; If r15 is not colSize, jump back to inner loop
```

```
75              mov rcx, offset newLine         ; 1st Parameter - new line character
76              sub rsp, 32                     ; Shadow Space
77              ;sub rsp 8                      ; If stack is not aligned
78              call printf                     ; Print a new line character
79              add rsp, 32                     ; or 40; Clean up stack
80
81              inc r14                         ; Increment row counter
82
83              ; 2nd row
84              mov r15, 0                      ; r15 = Column counter
85              ; Assumes the column size is divisible by 4
86              printInner2:
87                  mov r13, myMatrix           ; The address of the matrix in memory
88                  ; 1st print
89                  ; printf(myString, rdx)
90                  mov rcx, offset myString    ; 1st Parameter - string
91                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
92                  sub rsp, 32                 ; Shadow Space
93                  ;sub rsp, 8                 ; If stack is not aligned
94                  call printf                 ; Print the matrix element
95                  add rsp, 32                 ; or 40; Clean up stack
96
97                  inc r12                     ; Increment array counter
98
99                  ; 2nd print
100                 ; printf(myString, rdx)
101                 mov rcx, offset myString    ; 1st Parameter - string
102                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
```

```
103                 sub rsp, 32                 ; Shadow Space
104                 ;sub rsp, 8                 ; If stack is not aligned
105                 call printf                 ; Print the matrix element
106                 add rsp, 32                 ; or 40; Clean up stack
107
108                 inc r12                     ; Increment array counter
109
110                 ; 3rd print
111                 ; printf(myString, rdx)
112                 mov rcx, offset myString    ; 1st Parameter - string
113                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
114                 sub rsp, 32                 ; Shadow Space
115                 ;sub rsp, 8                 ; If stack is not aligned
116                 call printf                 ; Print the matrix element
117                 add rsp, 32                 ; or 40; Clean up stack
118
119                 inc r12                     ; Increment array counter
120
121                 ; 4th print
122                 ; printf(myString, rdx)
123                 mov rcx, offset myString    ; 1st Parameter - string
124                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
125                 sub rsp, 32                 ; Shadow Space
126                 ;sub rsp, 8                 ; If stack is not aligned
127                 call printf                 ; Print the matrix element
128                 add rsp, 32                 ; or 40; Clean up stack
```

```
130                 inc r12                     ; Increment array counter
131                 add r15, 4                  ; Increment column counter
132                 cmp r15, colSize            ; Compare column counter to colSize
133                 jnz printInner2             ; If r15 is not colSize, jump back to inner loop
134
135             mov rcx, offset newLine         ; 1st Parameter - new line character
136             sub rsp, 32                     ; Shadow Space
137             ;sub rsp 8                      ; If stack is not aligned
138             call printf                     ; Print a new line character
139             add rsp, 32                     ; or 40; Clean up stack
140
141             inc r14                         ; Increment row counter
142
143             ; 3rd row
144             mov r15, 0                      ; r15 = Column counter
145             ; Assumes the column size is divisible by 4
146             printInner3:
147                 mov r13, myMatrix           ; The address of the matrix in memory
148                 ; 1st print
149                 ; printf(myString, rdx)
150                 mov rcx, offset myString    ; 1st Parameter - string
151                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
152                 sub rsp, 32                 ; Shadow Space
153                 ;sub rsp, 8                 ; If stack is not aligned
154                 call printf                 ; Print the matrix element
155                 add rsp, 32                 ; or 40; Clean up stack
156
157                 inc r12                     ; Increment array counter
```

```
159                 ; 2nd print
160                 ; printf(myString, rdx)
161                 mov rcx, offset myString    ; 1st Parameter - string
162                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
163                 sub rsp, 32                 ; Shadow Space
164                 ;sub rsp, 8                 ; If stack is not aligned
165                 call printf                 ; Print the matrix element
166                 add rsp, 32                 ; or 40; Clean up stack
167
168                 inc r12                     ; Increment array counter
169
170                 ; 3rd print
171                 ; printf(myString, rdx)
172                 mov rcx, offset myString    ; 1st Parameter - string
173                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
174                 sub rsp, 32                 ; Shadow Space
175                 ;sub rsp, 8                 ; If stack is not aligned
176                 call printf                 ; Print the matrix element
177                 add rsp, 32                 ; or 40; Clean up stack
178
179                 inc r12                     ; Increment array counter
180
181                 ; 4th print
182                 ; printf(myString, rdx)
183                 mov rcx, offset myString    ; 1st Parameter - string
184                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
185                 sub rsp, 32                 ; Shadow Space
186                 ;sub rsp, 8                 ; If stack is not aligned
```

```
187                 call printf                 ; Print the matrix element
188                 add rsp, 32                 ; or 40; Clean up stack
189
190                 inc r12                     ; Increment array counter
191                 add r15, 4                  ; Increment column counter
192                 cmp r15, colSize            ; Compare column counter to colSize
193                 jnz printInner3             ; If r15 is not colSize, jump back to inner loop
194
195             mov rcx, offset newLine         ; 1st Parameter - new line character
196             sub rsp, 32                     ; Shadow Space
197             ;sub rsp 8                      ; If stack is not aligned
198             call printf                     ; Print a new line character
199             add rsp, 32                     ; or 40; Clean up stack
200
201             inc r14                         ; Increment row counter
202
203             ; 4th row
204             mov r15, 0                      ; r15 = Column counter
205             ; Assumes the column size is divisible by 4
206             printInner4:
207                 mov r13, myMatrix           ; The address of the matrix in memory
208                 ; 1st print
209                 ; printf(myString, rdx)
210                 mov rcx, offset myString    ; 1st Parameter - string
211                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
212                 sub rsp, 32                 ; Shadow Space
213                 ;sub rsp, 8                 ; If stack is not aligned
```

```
214                 call printf                 ; Print the matrix element
215                 add rsp, 32                 ; or 40; Clean up stack
216
217                 inc r12                     ; Increment array counter
218
219                 ; 2nd print
220                 ; printf(myString, rdx)
221                 mov rcx, offset myString    ; 1st Parameter - string
222                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
223                 sub rsp, 32                 ; Shadow Space
224                 ;sub rsp, 8                 ; If stack is not aligned
225                 call printf                 ; Print the matrix element
226                 add rsp, 32                 ; or 40; Clean up stack
227
228                 inc r12                     ; Increment array counter
229
230                 ; 3rd print
231                 ; printf(myString, rdx)
232                 mov rcx, offset myString    ; 1st Parameter - string
233                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
234                 sub rsp, 32                 ; Shadow Space
235                 ;sub rsp, 8                 ; If stack is not aligned
236                 call printf                 ; Print the matrix element
237                 add rsp, 32                 ; or 40; Clean up stack
238
239                 inc r12                     ; Increment array counter
240
241                 ; 4th print
242                 ; printf(myString, rdx)
243                 mov rcx, offset myString    ; 1st Parameter - string
```

```
244                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
245                 sub rsp, 32                 ; Shadow Space
246                 ;sub rsp, 8                 ; If stack is not aligned
247                 call printf                 ; Print the matrix element
248                 add rsp, 32                 ; or 40; Clean up stack
249
250                 inc r12                     ; Increment array counter
251                 add r15, 4                  ; Increment column counter
252                 cmp r15, colSize            ; Compare column counter to colSize
253                 jnz printInner4             ; If r15 is not colSize, jump back to inner loop
254
255             mov rcx, offset newLine         ; 1st Parameter - new line character
256             sub rsp, 32                     ; Shadow Space
257             ;sub rsp 8                      ; If stack is not aligned
258             call printf                     ; Print a new line character
259             add rsp, 32                     ; or 40; Clean up stack
260
261             inc r14                         ; Increment row counter
262             cmp r14, rowSize                ; Compare row counter to rowSize
263             jnz printOuter                  ; If r14 is not rowSize, jump back to outer loop
```

Optimization 2 – Loop Unrolling and Function Inlining

Taking my code from optimization 1, I implemented function inlining for the printMatrix function and put the code in main. I got the following results after running my program 3 times.

6,483 ms	**6,459 ms**	**6,484 ms**	**Average: 6,475.33 ms**	**Speed Increase: ~ 0.5%**

```
63          ; Print the matrix in a nice form
64          mov r12, 0                          ; r12 = Array counter
65          mov r14, 0                          ; r14 = Row counter
66          ; Assumes the row size is divisible by 4
67          printOuter1:
68              mov r15, 0                      ; r15 = Column counter
69              ; Assumes the column size is divisible by 4
70              printInner1:
71                  mov r13, myMatrix           ; The address of the matrix in memory
72                  ; 1st print
73                  ; printf(myString, rdx)
74                  mov rcx, offset myString    ; 1st Parameter - string
75                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
76                  sub rsp, 32                 ; Shadow Space
77                  sub rsp, 8                  ; If stack is not aligned
78                  call printf                 ; Print the matrix element
79                  add rsp, 40                 ; Clean up stack
80
81                  inc r12                     ; Increment array counter
82
83                  ; 2nd print
84                  ; printf(myString, rdx)
85                  mov rcx, offset myString    ; 1st Parameter - string
86                  mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
87                  sub rsp, 32                 ; Shadow Space
88                  sub rsp, 8                  ; If stack is not aligned
89                  call printf                 ; Print the matrix element
90                  add rsp, 40                 ; Clean up stack
```

```
92              inc r12                     ; Increment array counter
93
94              ; 3rd print
95              ; printf(myString, rdx)
96              mov rcx, offset myString    ; 1st Parameter - string
97              mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
98              sub rsp, 32                 ; Shadow Space
99              sub rsp, 8                  ; If stack is not aligned
100             call printf                 ; Print the matrix element
101             add rsp, 40                 ; Clean up stack
102
103             inc r12                     ; Increment array counter
104
105             ; 4th print
106             ; printf(myString, rdx)
107             mov rcx, offset myString    ; 1st Parameter - string
108             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
109             sub rsp, 32                 ; Shadow Space
110             sub rsp, 8                  ; If stack is not aligned
111             call printf                 ; Print the matrix element
112             add rsp, 40                 ; Clean up stack
113
114             inc r12                     ; Increment array counter
115             add r15, 4                  ; Increment column counter
116             cmp r15, colSize            ; Compare column counter to colSize
117             jnz printInner1             ; If r15 is not colSize, jump back to inner loop
118
119         mov rcx, offset newLine         ; 1st Parameter - new line character
120         sub rsp, 32                     ; Shadow Space
```

```
121         sub rsp, 8                      ; If stack is not aligned
122         call printf                     ; Print a new line character
123         add rsp, 40                     ; Clean up stack
124
125         inc r14                         ; Increment row counter
126
127         ; 2nd row
128         mov r15, 0                      ; r15 = Column counter
129         ; Assumes the column size is divisible by 4
130         printInner2:
131             mov r13, myMatrix           ; The address of the matrix in memory
132             ; 1st print
133             ; printf(myString, rdx)
134             mov rcx, offset myString    ; 1st Parameter - string
135             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
136             sub rsp, 32                 ; Shadow Space
137             sub rsp, 8                  ; If stack is not aligned
138             call printf                 ; Print the matrix element
139             add rsp, 40                 ; Clean up stack
140
141             inc r12                     ; Increment array counter
142
143             ; 2nd print
144             ; printf(myString, rdx)
145             mov rcx, offset myString    ; 1st Parameter - string
146             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
147             sub rsp, 32                 ; Shadow Space
148             sub rsp, 8                  ; If stack is not aligned
149             call printf                 ; Print the matrix element
150             add rsp, 40                 ; Clean up stack
```

```
152             inc r12                     ; Increment array counter
153
154             ; 3rd print
155             ; printf(myString, rdx)
156             mov rcx, offset myString    ; 1st Parameter - string
157             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
158             sub rsp, 32                 ; Shadow Space
159             sub rsp, 8                  ; If stack is not aligned
160             call printf                 ; Print the matrix element
161             add rsp, 40                 ; Clean up stack
162
163             inc r12                     ; Increment array counter
164
165             ; 4th print
166             ; printf(myString, rdx)
167             mov rcx, offset myString    ; 1st Parameter - string
168             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
169             sub rsp, 32                 ; Shadow Space
170             sub rsp, 8                  ; If stack is not aligned
171             call printf                 ; Print the matrix element
172             add rsp, 40                 ; Clean up stack
173
174             inc r12                     ; Increment array counter
175             add r15, 4                  ; Increment column counter
176             cmp r15, colSize            ; Compare column counter to colSize
177             jnz printInner2             ; If r15 is not colSize, jump back to inner loop
178
179         mov rcx, offset newLine         ; 1st Parameter - new line character
180         sub rsp, 32                     ; Shadow Space
```

```
181         sub rsp, 8                      ; If stack is not aligned
182         call printf                     ; Print a new line character
183         add rsp, 40                     ; Clean up stack
184
185         inc r14                         ; Increment row counter
186
187         ; 3rd row
188         mov r15, 0                      ; r15 = Column counter
189         ; Assumes the column size is divisible by 4
190         printInner3:
191             mov r13, myMatrix           ; The address of the matrix in memory
192             ; 1st print
193             ; printf(myString, rdx)
194             mov rcx, offset myString    ; 1st Parameter - string
195             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
196             sub rsp, 32                 ; Shadow Space
197             sub rsp, 8                  ; If stack is not aligned
198             call printf                 ; Print the matrix element
199             add rsp, 40                 ; Clean up stack
200
201             inc r12                     ; Increment array counter
202
203             ; 2nd print
204             ; printf(myString, rdx)
205             mov rcx, offset myString    ; 1st Parameter - string
206             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
207             sub rsp, 32                 ; Shadow Space
208             sub rsp, 8                  ; If stack is not aligned
209             call printf                 ; Print the matrix element
210             add rsp, 40                 ; Clean up stack
```

```
212             inc r12                     ; Increment array counter
213
214             ; 3rd print
215             ; printf(myString, rdx)
216             mov rcx, offset myString    ; 1st Parameter - string
217             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
218             sub rsp, 32                 ; Shadow Space
219             sub rsp, 8                  ; If stack is not aligned
220             call printf                 ; Print the matrix element
221             add rsp, 40                 ; Clean up stack
222
223             inc r12                     ; Increment array counter
224
225             ; 4th print
226             ; printf(myString, rdx)
227             mov rcx, offset myString    ; 1st Parameter - string
228             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
229             sub rsp, 32                 ; Shadow Space
230             sub rsp, 8                  ; If stack is not aligned
231             call printf                 ; Print the matrix element
232             add rsp, 40                 ; Clean up stack
233
234             inc r12                     ; Increment array counter
235             add r15, 4                  ; Increment column counter
236             cmp r15, colSize            ; Compare column counter to colSize
237             jnz printInner3             ; If r15 is not colSize, jump back to inner loop
238
239         mov rcx, offset newLine         ; 1st Parameter - new line character
240         sub rsp, 32                     ; Shadow Space
```

```
241         sub rsp, 8                      ; If stack is not aligned
242         call printf                     ; Print a new line character
243         add rsp, 40                     ; Clean up stack
244
245         inc r14                         ; Increment row counter
246
247         ; 4th row
248         mov r15, 0                      ; r15 = Column counter
249         ; Assumes the column size is divisible by 4
250         printInner4:
251             mov r13, myMatrix           ; The address of the matrix in memory
252             ; 1st print
253             ; printf(myString, rdx)
254             mov rcx, offset myString    ; 1st Parameter - string
255             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
256             sub rsp, 32                 ; Shadow Space
257             sub rsp, 8                  ; If stack is not aligned
258             call printf                 ; Print the matrix element
259             add rsp, 40                 ; Clean up stack
260
261             inc r12                     ; Increment array counter
262
263             ; 2nd print
264             ; printf(myString, rdx)
265             mov rcx, offset myString    ; 1st Parameter - string
266             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
267             sub rsp, 32                 ; Shadow Space
268             sub rsp, 8                  ; If stack is not aligned
269             call printf                 ; Print the matrix element
270             add rsp, 40                 ; Clean up stack
```

```
272             inc r12                     ; Increment array counter
273
274             ; 3rd print
275             ; printf(myString, rdx)
276             mov rcx, offset myString    ; 1st Parameter - string
277             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
278             sub rsp, 32                 ; Shadow Space
279             sub rsp, 8                  ; If stack is not aligned
280             call printf                 ; Print the matrix element
281             add rsp, 40                 ; Clean up stack
282
283             inc r12                     ; Increment array counter
284
285             ; 4th print
286             ; printf(myString, rdx)
287             mov rcx, offset myString    ; 1st Parameter - string
288             mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
289             sub rsp, 32                 ; Shadow Space
290             sub rsp, 8                  ; If stack is not aligned
291             call printf                 ; Print the matrix element
292             add rsp, 40                 ; Clean up stack
293
294             inc r12                     ; Increment array counter
295             add r15, 4                  ; Increment column counter
296             cmp r15, colSize            ; Compare column counter to colSize
297             jnz printInner4             ; If r15 is not colSize, jump back to inner loop
298
299         mov rcx, offset newLine         ; 1st Parameter - new line character
300         sub rsp, 32                     ; Shadow Space
```

```
301             sub rsp, 8                      ; If stack is not aligned
302             call printf                     ; Print a new line character
303             add rsp, 40                     ; Clean up stack
304
305             inc r14                         ; Increment row counter
306             cmp r14, rowSize                ; Compare row counter to rowSize
307             jnz printOuter1                 ; If r14 is not rowSize, jump back to outer loop
308
309         call flippyFlop                     ; Flippyflop the matrix
310
311         mov rcx, offset transposePrompt     ; 1st Parameter - Labeling the transpose matrix
312         sub rsp, 32                         ; Shadow Space
313         sub rsp, 8                          ; Stack is not aligned
314         call printf                         ; Print the label
315         add rsp, 40                         ; Clean up stack
316
317         ; Print the matrix in a nice form
318         mov r12, 0                          ; r12 = Array counter
319         mov r14, 0                          ; r14 = Row counter
320         ; Assumes the row size is divisible by 4
321         printOuter2:
322             mov r15, 0                      ; r15 = Column counter
323             ; Assumes the column size is divisible by 4
324             printInner5:
325                 mov r13, myMatrix           ; The address of the matrix in memory
326                 ; 1st print
327                 ; printf(myString, rdx)
328                 mov rcx, offset myString    ; 1st Parameter - string
329                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
330                 sub rsp, 32                 ; Shadow Space
```

```
331                 sub rsp, 8                  ; If stack is not aligned
332                 call printf                 ; Print the matrix element
333                 add rsp, 40                 ; Clean up stack
334
335                 inc r12                     ; Increment array counter
336
337                 ; 2nd print
338                 ; printf(myString, rdx)
339                 mov rcx, offset myString    ; 1st Parameter - string
340                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
341                 sub rsp, 32                 ; Shadow Space
342                 sub rsp, 8                  ; If stack is not aligned
343                 call printf                 ; Print the matrix element
344                 add rsp, 40                 ; Clean up stack
345
346                 inc r12                     ; Increment array counter
347
348                 ; 3rd print
349                 ; printf(myString, rdx)
350                 mov rcx, offset myString    ; 1st Parameter - string
351                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
352                 sub rsp, 32                 ; Shadow Space
353                 sub rsp, 8                  ; If stack is not aligned
354                 call printf                 ; Print the matrix element
355                 add rsp, 40                 ; Clean up stack
356
357                 inc r12                     ; Increment array counter
358
359                 ; 4th print
360                 ; printf(myString, rdx)
```

```
361                 mov rcx, offset myString    ; 1st Parameter - string
362                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
363                 sub rsp, 32                 ; Shadow Space
364                 sub rsp, 8                  ; If stack is not aligned
365                 call printf                 ; Print the matrix element
366                 add rsp, 40                 ; Clean up stack
367
368                 inc r12                     ; Increment array counter
369                 add r15, 4                  ; Increment column counter
370                 cmp r15, colSize            ; Compare column counter to colSize
371                 jnz printInner5             ; If r15 is not colSize, jump back to inner loop
372
373             mov rcx, offset newLine         ; 1st Parameter - new line character
374             sub rsp, 32                     ; Shadow Space
375             sub rsp, 8                      ; If stack is not aligned
376             call printf                     ; Print a new line character
377             add rsp, 40                     ; Clean up stack
378
379             inc r14                         ; Increment row counter
380
381             ; 2nd row
382             mov r15, 0                      ; r15 = Column counter
383             ; Assumes the column size is divisible by 4
384             printInner6:
385                 mov r13, myMatrix           ; The address of the matrix in memory
386                 ; 1st print
387                 ; printf(myString, rdx)
388                 mov rcx, offset myString    ; 1st Parameter - string
389                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
390                 sub rsp, 32                 ; Shadow Space
```

```
391                 sub rsp, 8                  ; If stack is not aligned
392                 call printf                 ; Print the matrix element
393                 add rsp, 40                 ; Clean up stack
394
395                 inc r12                     ; Increment array counter
396
397                 ; 2nd print
398                 ; printf(myString, rdx)
399                 mov rcx, offset myString    ; 1st Parameter - string
400                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
401                 sub rsp, 32                 ; Shadow Space
402                 sub rsp, 8                  ; If stack is not aligned
403                 call printf                 ; Print the matrix element
404                 add rsp, 40                 ; Clean up stack
405
406                 inc r12                     ; Increment array counter
407
408                 ; 3rd print
409                 ; printf(myString, rdx)
410                 mov rcx, offset myString    ; 1st Parameter - string
411                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
412                 sub rsp, 32                 ; Shadow Space
413                 sub rsp, 8                  ; If stack is not aligned
414                 call printf                 ; Print the matrix element
415                 add rsp, 40                 ; Clean up stack
416
417                 inc r12                     ; Increment array counter
418
419                 ; 4th print
420                 ; printf(myString, rdx)
```

```
421                 mov rcx, offset myString    ; 1st Parameter - string
422                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
423                 sub rsp, 32                 ; Shadow Space
424                 sub rsp, 8                  ; If stack is not aligned
425                 call printf                 ; Print the matrix element
426                 add rsp, 40                 ; Clean up stack
427
428                 inc r12                     ; Increment array counter
429                 add r15, 4                  ; Increment column counter
430                 cmp r15, colSize            ; Compare column counter to colSize
431                 jnz printInner6             ; If r15 is not colSize, jump back to inner loop
432
433             mov rcx, offset newLine         ; 1st Parameter - new line character
434             sub rsp, 32                     ; Shadow Space
435             sub rsp, 8                      ; If stack is not aligned
436             call printf                     ; Print a new line character
437             add rsp, 40                     ; Clean up stack
438
439             inc r14                         ; Increment row counter
440
441             ; 3rd row
442             mov r15, 0                      ; r15 = Column counter
443             ; Assumes the column size is divisible by 4
444             printInner7:
445                 mov r13, myMatrix           ; The address of the matrix in memory
446                 ; 1st print
447                 ; printf(myString, rdx)
448                 mov rcx, offset myString    ; 1st Parameter - string
449                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
450                 sub rsp, 32                 ; Shadow Space
```

```
451                 sub rsp, 8                  ; If stack is not aligned
452                 call printf                 ; Print the matrix element
453                 add rsp, 40                 ; Clean up stack
454
455                 inc r12                     ; Increment array counter
456
457                 ; 2nd print
458                 ; printf(myString, rdx)
459                 mov rcx, offset myString    ; 1st Parameter - string
460                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
461                 sub rsp, 32                 ; Shadow Space
462                 sub rsp, 8                  ; If stack is not aligned
463                 call printf                 ; Print the matrix element
464                 add rsp, 40                 ; Clean up stack
465
466                 inc r12                     ; Increment array counter
467
468                 ; 3rd print
469                 ; printf(myString, rdx)
470                 mov rcx, offset myString    ; 1st Parameter - string
471                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
472                 sub rsp, 32                 ; Shadow Space
473                 sub rsp, 8                  ; If stack is not aligned
474                 call printf                 ; Print the matrix element
475                 add rsp, 40                 ; Clean up stack
476
477                 inc r12                     ; Increment array counter
478
479                 ; 4th print
480                 ; printf(myString, rdx)
```

```
481                 mov rcx, offset myString    ; 1st Parameter - string
482                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
483                 sub rsp, 32                 ; Shadow Space
484                 sub rsp, 8                  ; If stack is not aligned
485                 call printf                 ; Print the matrix element
486                 add rsp, 40                 ; Clean up stack
487
488                 inc r12                     ; Increment array counter
489                 add r15, 4                  ; Increment column counter
490                 cmp r15, colSize            ; Compare column counter to colSize
491                 jnz printInner7             ; If r15 is not colSize, jump back to inner loop
492
493             mov rcx, offset newLine         ; 1st Parameter - new line character
494             sub rsp, 32                     ; Shadow Space
495             sub rsp, 8                      ; If stack is not aligned
496             call printf                     ; Print a new line character
497             add rsp, 40                     ; Clean up stack
498
499             inc r14                         ; Increment row counter
500
501             ; 4th row
502             mov r15, 0                      ; r15 = Column counter
503             ; Assumes the column size is divisible by 4
504             printInner8:
505                 mov r13, myMatrix           ; The address of the matrix in memory
506                 ; 1st print
507                 ; printf(myString, rdx)
508                 mov rcx, offset myString    ; 1st Parameter - string
509                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
510                 sub rsp, 32                 ; Shadow Space
```

```
511                 sub rsp, 8                  ; If stack is not aligned
512                 call printf                 ; Print the matrix element
513                 add rsp, 40                 ; Clean up stack
514
515                 inc r12                     ; Increment array counter
516
517                 ; 2nd print
518                 ; printf(myString, rdx)
519                 mov rcx, offset myString    ; 1st Parameter - string
520                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
521                 sub rsp, 32                 ; Shadow Space
522                 sub rsp, 8                  ; If stack is not aligned
523                 call printf                 ; Print the matrix element
524                 add rsp, 40                 ; Clean up stack
525
526                 inc r12                     ; Increment array counter
527
528                 ; 3rd print
529                 ; printf(myString, rdx)
530                 mov rcx, offset myString    ; 1st Parameter - string
531                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
532                 sub rsp, 32                 ; Shadow Space
533                 sub rsp, 8                  ; If stack is not aligned
534                 call printf                 ; Print the matrix element
535                 add rsp, 40                 ; Clean up stack
536
537                 inc r12                     ; Increment array counter
538
539                 ; 4th print
540                 ; printf(myString, rdx)
```

```
541                 mov rcx, offset myString    ; 1st Parameter - string
542                 mov rdx, [r13 + 8*r12]      ; 2nd Parameter - array value
543                 sub rsp, 32                 ; Shadow Space
544                 sub rsp, 8                  ; If stack is not aligned
545                 call printf                 ; Print the matrix element
546                 add rsp, 40                 ; Clean up stack
547
548                 inc r12                     ; Increment array counter
549                 add r15, 4                  ; Increment column counter
550                 cmp r15, colSize            ; Compare column counter to colSize
551                 jnz printInner8             ; If r15 is not colSize, jump back to inner loop
552
553             mov rcx, offset newLine         ; 1st Parameter - new line character
554             sub rsp, 32                     ; Shadow Space
555             sub rsp, 8                      ; If stack is not aligned
556             call printf                     ; Print a new line character
557             add rsp, 40                     ; Clean up stack
558
559             inc r14                         ; Increment row counter
560             cmp r14, rowSize                ; Compare row counter to rowSize
561             jnz printOuter2                 ; If r14 is not rowSize, jump back to outer loop
562
563         mov rcx, 0                          ; Exit code of 0
564         call ExitProcess
```