I/O programming in C (interrupts/polling) = 15 points
Bus arbitration waveforms (daisy chain) = 10 points
FSM (state diagrams + output waveforms) = 10 points
RM/EDF real-time task scheduling = 5 points
Cache mapping (direct-mapped/fully-associative/set-associative) = 20 points
Virtual memory (page fault rates) = 15 points
Floating-point arithmetic (conversion between floating-point and decimal formats, add/subtract) = 10 points
Pipelining (NOP insertion for data hazards) = 5 points

---

double X[256][256]
   a) 2x2kb
   b) 1x4kb

```
for(i=0;i<N;i++){
      Double sum = 0;
      for (j=0;j<N;j++){
            sum = sum + X[i][j];
      }
      for(j=0;j<N;j++){
            X[i][j] = X[i][j]/sum;
      }
}
```

Rate = # access faulting / total # accesses
256*(256*1 + 2*256) total accesses
Double is 8bytes/double * 256/row = 2kb
   a) First access is X[0][0] and is a fault, when the 2kb page is loaded, X[0][1] etc are all hits for the whole row. First loop has 1 hit, 255 hits. Second loop has 256 hits. When i becomes 1, same as when 0; first is a fault, rest are hit for the whole row.
      Therefore, 1x256 page faults (one per row).
   b) First access is X[0][0] but loads both i=0 and i=1. Therefore 1x256/2 page faults

---

```
+3.75 = 11.11*2^0 (2^1,2^0,2^-1,2^-2) = 1.111 * 2^1 -> 1+127 = 128
0       1000 0000     11100….
Sign     E-127=1     decimal

-0.75 = 0.11 * 2^0 = 1.1 * 2^-1 -> -1 +127 = 126
1     0111 1110     10000….
     E-127 = -1

-2^-128 = -1.00 * 2^128  -> -128 + 127 = -1 (hitting normalization!)
Change to=> 0.01 * 2^-126 (normalized case)
E= 0000 0000   0    01000…
            Implicit
```

```
Z= X - Y = X - (-Y) = X + “-Y”
X = 0 0111 1111 1100...
Y = 1 0111 1101 0000…
-Y= 0 0111 1101 0000…
Z = 0 1000 0000 1 0000…
Z = +2^128-127 = +2
```

```
ADD R4,R0,R2  //R2 is destination
ADD R4,(R0)
MOV (R2),R0
MOV R0,(R4)
MOV R4,(R1)
MOV R2,(R0)
ADD #4,R0,R0
ADD #4,R1,R1
ADD #4,R2,R2
ADD R0,R4,R4
```

```
<<see next page>> hand written
M2 -(R2)-> A
A  -(G2)-> M2
M2 -> Busy
M1 -(R1)-> A
A  -(G1)-> M1
M1 -> Busy
Busy -> A
M3 -(R3)->A
A  -(G3)-> M3
M3 -> Busy
```

200 - 0010 0000 00|00
210 - 0010 0001 00|00
090 - 0000 1001 00|00
080 - 0000 1000 00|00

Miss - hit - miss - hit

	111	110	101	100	011	010	001	000	
0010	21c	218	214	210	20c	208	204	200	000
									001
									010
									011
	09c	098	094	090	08c	088	084	080	100
									101
									110
									111

2way set associative: 090 - (0000 1<tag>)(00<set>)(1 00<word>)|00
Miss - hit - miss - hit
Add: 488 - 0100 1000 1000 <replaces first row (of the 200s)> is miss

	111	110	101	100	011	010	001	000		sets
0010	21c	218	214	210	20c	208	204	200		00
	09c	098	094	090	08c	088	084	080		00
										01
										01
										10
										10
										11
										11

Fully associative cache:
Miss - hit - miss - hit - miss

	111	110	101	100	011	010	001	000		
0010	21c	218	214	210	20c	208	204	200		
	09c	098	094	090	08c	088	084	080		

4way set associative: 090 - (0000 10<tag>)(0<set>)(1 00<word>)|00
Miss - hit - miss - hit
Add: 488 - 0100 1000 1000 <replaces first row (of the 200s)> is miss

	111	110	101	100	011	010	001	000		sets
0010	21c	218	214	210	20c	208	204	200		0
	09c	098	094	090	08c	088	084	080		0
	49c	498	494	490	48c	488	484	480		0
										0
										1
										1
										1
										1