In [6]:
```
import pandas as pd
# Displaying the TestInfo.csv as dataframe for knowing more insig
df1 = pd.read_csv('TestInfo.csv')
```

In [2]:
```
df1
```

Out[2]:

	TestId	Device	CPUFrequency (MHz)	Threads	MLNetwork
0	17	Device_0	1000	5	AlexNet
1	16	Device_0	1000	5	AlexNet
2	39	Device_1	1000	3	AlexNet
3	31	Device_0	1000	5	AlexNet
4	30	Device_0	1000	4	AlexNet
5	25	Device_0	2000	5	AlexNet
6	45	Device_0	1000	4	MobileNet
7	34	Device_0	2000	3	AlexNet
8	6	Device_0	1000	5	MobileNet
9	27	Device_0	1000	1	AlexNet
10	46	Device_0	1000	5	MobileNet
11	33	Device_0	2000	2	AlexNet
12	24	Device_0	2000	5	AlexNet
13	4	Device_0	1000	5	MobileNet
14	32	Device_0	2000	1	AlexNet
15	2	Device_0	1000	5	MobileNet
16	37	Device_1	1000	1	AlexNet
17	20	Device_0	2000	5	AlexNet
18	23	Device_0	2000	5	AlexNet
19	12	Device_0	1000	5	AlexNet
20	1	Device_0	1000	5	MobileNet
21	0	Device_0	1000	5	MobileNet
22	21	Device_0	2000	5	AlexNet
23	44	Device_0	1000	3	MobileNet
24	29	Device_0	1000	3	AlexNet

	TestId	Device	CPUFrequency (MHz)	Threads	MLNetwork
25	13	Device_0	1000	5	AlexNet
26	7	Device_0	1000	5	MobileNet
27	8	Device_0	1000	5	MobileNet
28	19	Device_0	2000	5	AlexNet
29	9	Device_0	1000	5	AlexNet
30	26	Device_0	2000	5	AlexNet
31	43	Device_0	1000	2	MobileNet
32	28	Device_0	1000	2	AlexNet
33	18	Device_0	2000	5	AlexNet
34	3	Device_0	1000	5	MobileNet
35	11	Device_0	1000	5	AlexNet
36	5	Device_0	1000	5	MobileNet
37	38	Device_1	1000	2	AlexNet
38	41	Device_1	1000	5	AlexNet
39	22	Device_0	2000	5	AlexNet
40	36	Device_0	2000	5	AlexNet
41	10	Device_0	1000	5	AlexNet
42	40	Device_1	1000	4	AlexNet
43	14	Device_0	1000	5	AlexNet
44	35	Device_0	2000	4	AlexNet
45	15	Device_0	1000	5	AlexNet
46	42	Device_0	1000	1	MobileNet

In [10]:
```
df1.describe()
```

Out[10]:

	TestId	CPUFrequency (MHz)	Threads
count	47.000000	47.000000	47.000000
mean	23.000000	1297.872340	4.148936
std	13.711309	462.267268	1.366982
min	0.000000	1000.000000	1.000000
25%	11.500000	1000.000000	3.500000

	TestId	CPUFrequency (MHz)	Threads
50%	23.000000	1000.000000	5.000000
75%	34.500000	2000.000000	5.000000
max	46.000000	2000.000000	5.000000

In [7]:
```
# Displaying the TestResults.pickle as dataframe for knowing more
df2 = pd.read_pickle('TestResults.pickle')
```

In [8]:
```
df2
```

Out[8]:

	TestId	Build	Optimised	Time (ms)	PeakMemory (MB)
0	33	10	N	125.000000	307
1	43	10	Y	50.000000	50
2	36	10	N	50.000000	458
3	1	2	N	64.000000	451
4	16	8	N	104.000000	453
5	27	10	N	500.000000	300
6	26	9	N	53.000000	457
7	41	10	N	200.000000	454
8	37	10	N	800.000000	298
9	46	10	Y	20.000000	86
10	4	5	N	63.000000	448
11	44	10	Y	33.333333	53
12	24	7	N	50.000000	458
13	9	1	N	101.000000	450
14	6	7	Y	25.000000	450
15	0	1	N	65.000000	450
16	22	5	N	51.000000	456
17	13	5	N	101.000000	452
18	29	10	N	166.666667	311
19	18	1	N	53.000000	456
20	17	9	N	102.000000	449

	TestId	Build	Optimised	Time (ms)	PeakMemory (MB)
21	45	10	Y	25.000000	85
22	21	4	N	50.000000	457
23	11	3	N	104.000000	453
24	19	2	N	53.000000	456
25	42	10	Y	100.000000	51
26	38	10	N	500.000000	305
27	8	9	Y	19.000000	449
28	40	10	N	250.000000	449
29	30	10	N	125.000000	450
30	31	10	N	100.000000	449
31	23	6	N	54.000000	458
32	20	3	N	52.000000	457
33	32	10	N	250.000000	301
34	12	4	N	102.000000	453
35	2	3	N	65.000000	450
36	35	10	N	300.000000	460
37	5	6	Y	21.000000	449
38	28	10	N	250.000000	310
39	25	8	N	53.000000	458
40	10	2	N	102.000000	449
41	39	10	N	333.333333	302
42	15	7	N	97.000000	455
43	34	10	N	83.333333	308
44	14	6	N	99.000000	450
45	3	4	N	66.000000	450
46	7	8	Y	23.000000	451

In [9]:
```
df2.describe()
```

Out[9]:

	TestId	Build	Time (ms)	PeakMemory (MB)
count	47.000000	47.000000	47.000000	47.000000

	TestId	Build	Time (ms)	PeakMemory (MB)
mean	23.000000	7.127660	128.801418	383.234043
std	13.711309	3.187046	149.035930	125.454168
min	0.000000	1.000000	19.000000	50.000000
25%	11.500000	4.500000	51.500000	309.000000
50%	23.000000	8.000000	83.333333	450.000000
75%	34.500000	10.000000	114.500000	454.500000
max	46.000000	10.000000	800.000000	460.000000

In [11]:
```
# Merging both dataframes based on common identifier 'TestId' for
df3 = pd.merge(df1, df2, on="TestId")
```

In [12]:
```
df3
```

Out[12]:

	TestId	Device	CPUFrequency (MHz)	Threads	MLNetwork	Build	Optimised	
0	17	Device_0	1000	5	AlexNet	9	N	1
1	16	Device_0	1000	5	AlexNet	8	N	1
2	39	Device_1	1000	3	AlexNet	10	N	3
3	31	Device_0	1000	5	AlexNet	10	N	1
4	30	Device_0	1000	4	AlexNet	10	N	1
5	25	Device_0	2000	5	AlexNet	8	N	
6	45	Device_0	1000	4	MobileNet	10	Y	
7	34	Device_0	2000	3	AlexNet	10	N	
8	6	Device_0	1000	5	MobileNet	7	Y	
9	27	Device_0	1000	1	AlexNet	10	N	5
10	46	Device_0	1000	5	MobileNet	10	Y	
11	33	Device_0	2000	2	AlexNet	10	N	1
12	24	Device_0	2000	5	AlexNet	7	N	
13	4	Device_0	1000	5	MobileNet	5	N	
14	32	Device_0	2000	1	AlexNet	10	N	2
15	2	Device_0	1000	5	MobileNet	3	N	

	TestId	Device	CPUFrequency (MHz)	Threads	MLNetwork	Build	Optimised	
16	37	Device_1	1000	1	AlexNet	10	N	8
17	20	Device_0	2000	5	AlexNet	3	N	
18	23	Device_0	2000	5	AlexNet	6	N	
19	12	Device_0	1000	5	AlexNet	4	N	1
20	1	Device_0	1000	5	MobileNet	2	N	
21	0	Device_0	1000	5	MobileNet	1	N	
22	21	Device_0	2000	5	AlexNet	4	N	
23	44	Device_0	1000	3	MobileNet	10	Y	
24	29	Device_0	1000	3	AlexNet	10	N	1
25	13	Device_0	1000	5	AlexNet	5	N	1
26	7	Device_0	1000	5	MobileNet	8	Y	
27	8	Device_0	1000	5	MobileNet	9	Y	
28	19	Device_0	2000	5	AlexNet	2	N	
29	9	Device_0	1000	5	AlexNet	1	N	1
30	26	Device_0	2000	5	AlexNet	9	N	
31	43	Device_0	1000	2	MobileNet	10	Y	
32	28	Device_0	1000	2	AlexNet	10	N	2
33	18	Device_0	2000	5	AlexNet	1	N	
34	3	Device_0	1000	5	MobileNet	4	N	
35	11	Device_0	1000	5	AlexNet	3	N	1
36	5	Device_0	1000	5	MobileNet	6	Y	
37	38	Device_1	1000	2	AlexNet	10	N	5
38	41	Device_1	1000	5	AlexNet	10	N	2
39	22	Device_0	2000	5	AlexNet	5	N	
40	36	Device_0	2000	5	AlexNet	10	N	
41	10	Device_0	1000	5	AlexNet	2	N	1
42	40	Device_1	1000	4	AlexNet	10	N	2
43	14	Device_0	1000	5	AlexNet	6	N	
44	35	Device_0	2000	4	AlexNet	10	N	3
45	15	Device_0	1000	5	AlexNet	7	N	

	TestId	Device	CPUFrequency (MHz)	Threads	MLNetwork	Build	Optimised	
46	42	Device_0	1000	1	MobileNet	10	Y	1

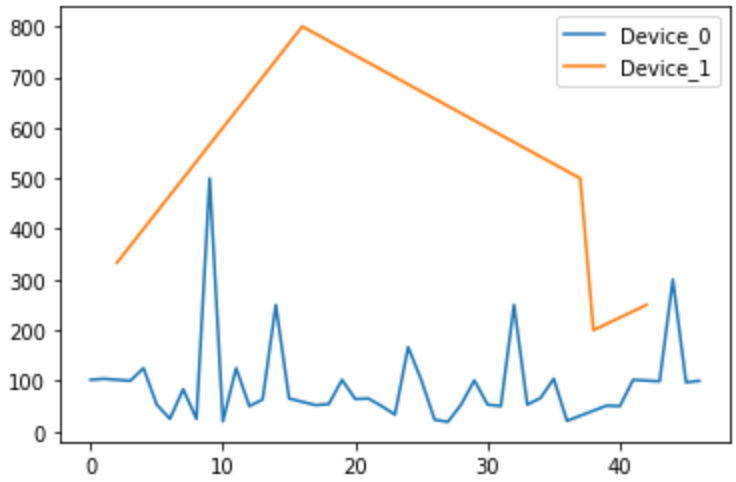
In [13]:
```
df3.describe()
```

Out[13]:

	TestId	CPUFrequency (MHz)	Threads	Build	Time (ms)	PeakMem (M
count	47.000000	47.000000	47.000000	47.000000	47.000000	47.000(
mean	23.000000	1297.872340	4.148936	7.127660	128.801418	383.234(
std	13.711309	462.267268	1.366982	3.187046	149.035930	125.454
min	0.000000	1000.000000	1.000000	1.000000	19.000000	50.000(
25%	11.500000	1000.000000	3.500000	4.500000	51.500000	309.000(
50%	23.000000	1000.000000	5.000000	8.000000	83.333333	450.000(
75%	34.500000	2000.000000	5.000000	10.000000	114.500000	454.500(
max	46.000000	2000.000000	5.000000	10.000000	800.000000	460.000(

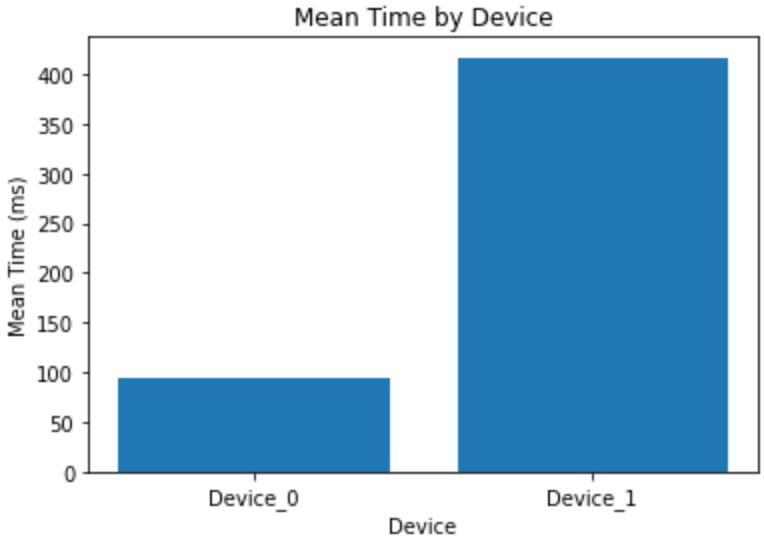
In [14]:
```
import matplotlib.pyplot as plt
```

In [34]:
```
#Line graph to illustrate the correlation between the time and de
#From the data points its evident that the Device_0 is being used
time_0 = df3[df3['Device'] == 'Device_0']['Time (ms)']
time_1 = df3[df3['Device'] == 'Device_1']['Time (ms)']
plt.plot(time_0, label='Device_0')
plt.plot(time_1, label='Device_1')
plt.legend()
plt.show()
```



In [23]:

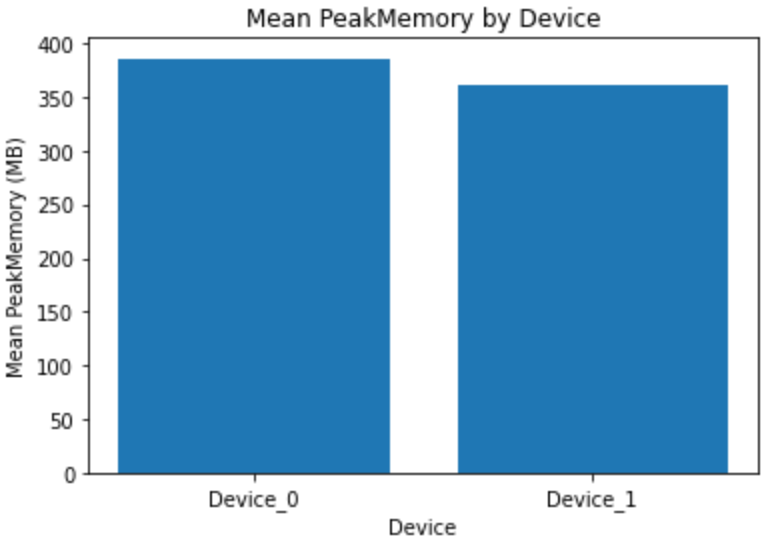
```
#The bar graph also support the fact that  higer time is consumin

# Grouping the data by Device and calculating the mean Time for e
device_time_means = df3.groupby('Device')['Time (ms)'].mean()

# Creating a bar plot
plt.bar(device_time_means.index, device_time_means.values)
plt.xlabel('Device')
plt.ylabel('Mean Time (ms)')
plt.title('Mean Time by Device')

# Displaying the plot
plt.show()
```



In [26]:

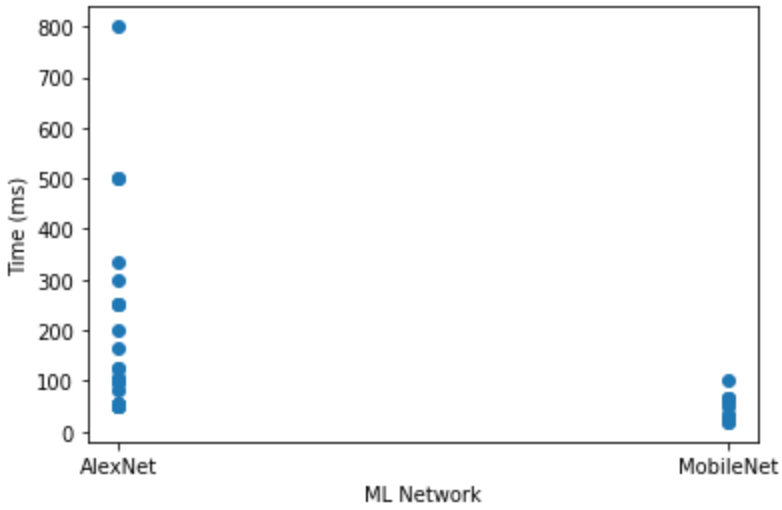
```
#The bar graph shows that higer peakmemory is ultised by the Devi

# Grouping the data by Device and calculating the mean PeakMemory
```

```
device_PeakMemory_means = df3.groupby('Device')['PeakMemory (MB)'

# Creating a bar plot
plt.bar(device_PeakMemory_means.index, device_PeakMemory_means.va
plt.xlabel('Device')
plt.ylabel('Mean PeakMemory (MB)')
plt.title('Mean PeakMemory by Device')

# Displaying the plot
plt.show()
```



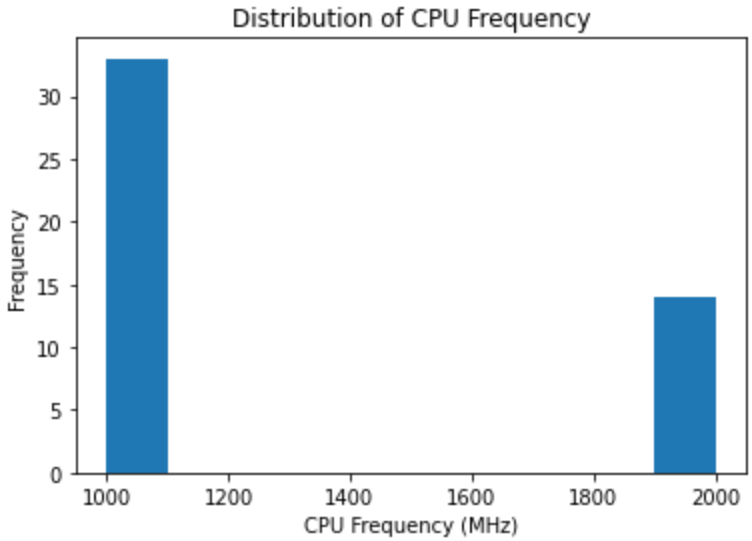
In [27]:

```
#The scatter plots illustrate that tests with MobileNet tend to h

# Extract the values from the 'Time' and 'MLNetwork' columns
time = df3['Time (ms)']
ml_network = df3['MLNetwork']

# Create a scatter plot
plt.scatter(ml_network, time)

# Add labels to the x and y axes
plt.xlabel('ML Network')
plt.ylabel('Time (ms)')

# Show the plot
plt.show()
```

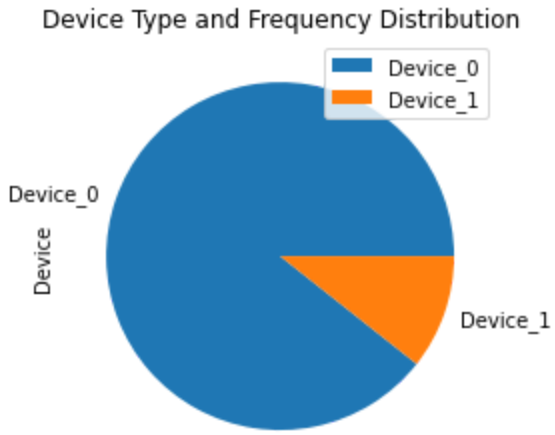


In [31]:

```
#Illustrates the Distribution of CPU Frequency and it is evident

# Set up the figure and axis
fig, ax = plt.subplots()

# Plot the histogram
ax.hist(df3['CPUFrequency (MHz)'], bins=10)

# Add a title and x- and y-axis labels
ax.set_title('Distribution of CPU Frequency')
ax.set_xlabel('CPU Frequency (MHz)')
ax.set_ylabel('Frequency')

# Show the plot
plt.show()
```



In [35]:
```
#Illustrate the Frequency disribution of Device Type

device = df3['Device']
frequency = df3['CPUFrequency (MHz)']

# Count the frequency of each device type
device_counts = device.value_counts()

# Plot the pie chart
device_counts.plot.pie(legend=True)

# Add a title
plt.title('Device Type and Frequency Distribution')

# Display the plot
plt.show()
```



In [42]:
```
# Tests with optimisation enabled tend to have shorter times and

# Grouping data by optimization flag
optimized = df3[df3['Optimised'] == 'Y']
not_optimized = df3[df3['Optimised'] == 'N']

# Calculating the mean time and memory for each group
optimized_time = optimized['Time (ms)'].mean()
not_optimized_time = not_optimized['Time (ms)'].mean()
optimized_memory = optimized['PeakMemory (MB)'].mean()
not_optimized_memory = not_optimized['PeakMemory (MB)'].mean()

# Set up bar plot
x = ['Optimized', 'Not Optimized']
y1 = [optimized_time, not_optimized_time]
y2 = [optimized_memory, not_optimized_memory]
```

```
fig, ax1 = plt.subplots()

# Plot time on left y-axis
color = 'tab:red'
ax1.set_xlabel('Optimization Flag')
ax1.set_ylabel('Time (ms)', color=color)
ax1.bar(x, y1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

# Plot memory on right y-axis
ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Peak Memory (MB)', color=color)
ax2.bar(x, y2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

# Show plot
plt.show()
```