In [1]:
```
import pandas as pd
import  numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv('CPU_r23_v2.csv')
```

In [3]:
```
df
```

Out[3]:

	manufacturer	cpuName	singleScore	multiScore	cores	threads	baseClock	turboClock	
0	AMD	Threadripper 3990X	1262	75671	64	128	2.9	4.50	[
1	AMD	Threadripper Pro 3995WX	1231	73220	64	128	2.7	4.20	[
2	AMD	Epyc 7702P	993	48959	64	128	2.0	3.35	[
3	AMD	Threadripper 3970X	1308	46874	32	64	3.7	4.50	[
4	AMD	Threadripper Pro 3975WX	1244	43450	32	64	3.5	4.20	[
...	...	...	...	...	...	...	...	...	
210	AMD	Ryzen 3 4300GE	1215	5798	4	8	3.5	4.00	[
211	Intel	Core i7 1185G7	1473	5783	4	8	1.2	4.80	
212	Intel	Core i7 11370H	1535	5778	4	8	3.0	4.80	
213	Intel	Core i3 10105F	1172	5776	4	8	3.7	4.40	[
214	AMD	Ryzen 3 3100	1105	5423	4	8	3.6	3.90	[

215 rows × 9 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   manufacturer  215 non-null    object 
 1   cpuName       215 non-null    object 
 2   singleScore   215 non-null    int64  
 3   multiScore    215 non-null    int64  
 4   cores         215 non-null    int64  
 5   threads       215 non-null    int64  
 6   baseClock     215 non-null    float64
 7   turboClock    215 non-null    float64
 8   type          215 non-null    object 
dtypes: float64(2), int64(4), object(3)
memory usage: 15.2+ KB
```

In [5]:
```
df.shape
```

Out[5]:
```
(215, 9)
```

In [6]:
```
df.describe()
```

Out[6]:

	singleScore	multiScore	cores	threads	baseClock	turboClock
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000
mean	1367.553488	12979.502326	9.367442	17.739535	3.053953	4.514651
std	239.829487	8905.742643	7.823393	15.829910	0.653009	0.441463
min	903.000000	5423.000000	4.000000	6.000000	1.100000	3.200000
25%	1207.000000	8186.500000	6.000000	12.000000	2.500000	4.200000
50%	1312.000000	10890.000000	8.000000	16.000000	3.200000	4.500000
75%	1534.000000	14393.500000	9.000000	16.000000	3.600000	4.800000
max	2082.000000	75671.000000	64.000000	128.000000	4.200000	5.500000

In [7]:
```
df.head()
```

Out[7]:

	manufacturer	cpuName	singleScore	multiScore	cores	threads	baseClock	turboClock	
0	AMD	Threadripper 3990X	1262	75671	64	128	2.9	4.50	Des
1	AMD	Threadripper Pro 3995WX	1231	73220	64	128	2.7	4.20	Des
2	AMD	Epyc 7702P	993	48959	64	128	2.0	3.35	Des
3	AMD	Threadripper 3970X	1308	46874	32	64	3.7	4.50	Des
4	AMD	Threadripper Pro 3975WX	1244	43450	32	64	3.5	4.20	Des

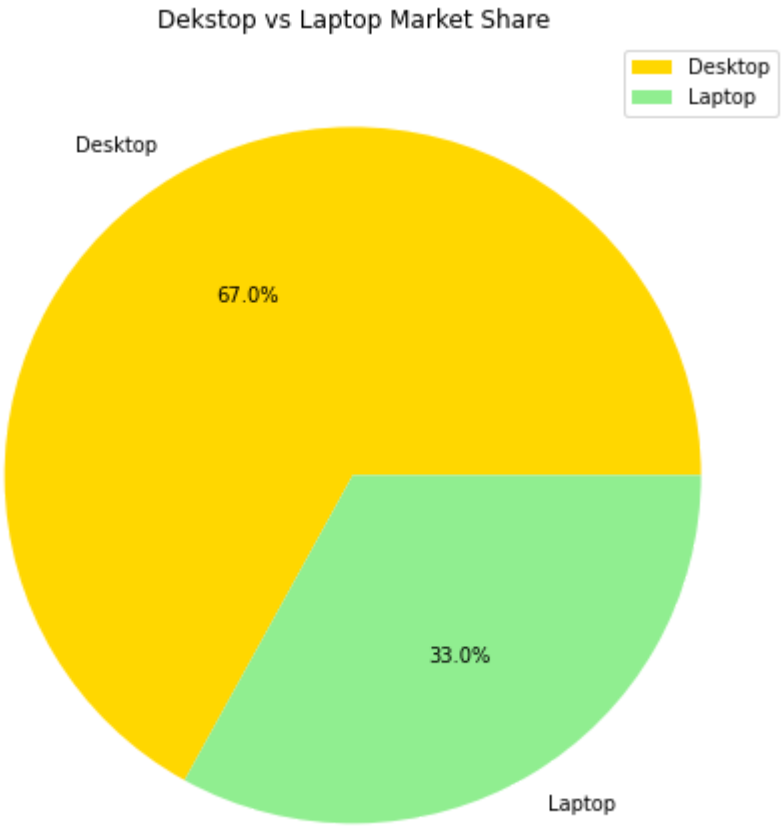
In [8]:
```
df.isnull().sum()
```

Out[8]:
```
manufacturer    0
cpuName         0
singleScore     0
multiScore      0
cores           0
threads         0
baseClock       0
turboClock      0
type            0
dtype: int64
```

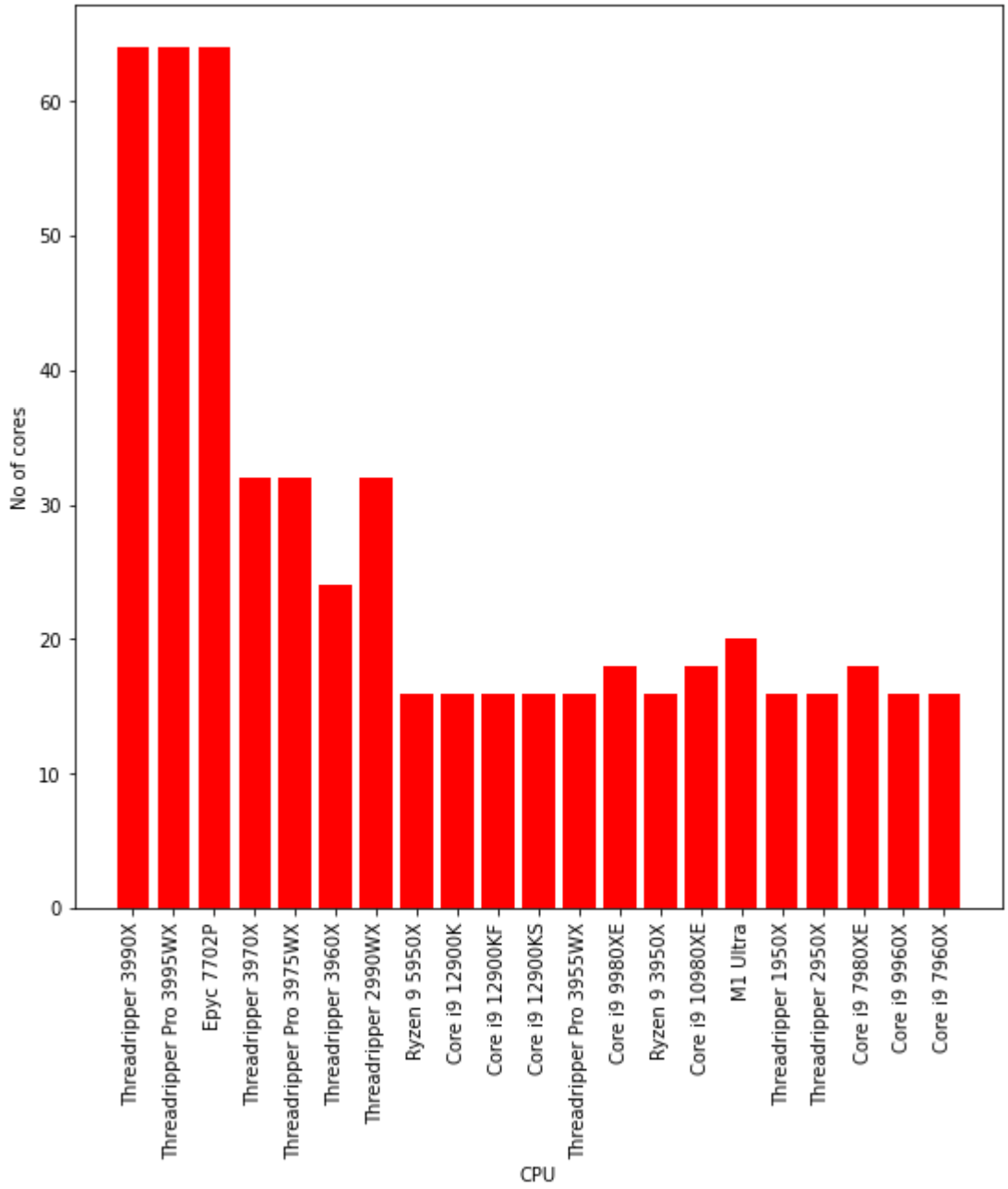
In [16]:
```
#1 What is the count of CPU types?
a=df['type'].value_counts()
a
```

Out[16]:
```
Desktop    144
Laptop      71
Name: type, dtype: int64
```

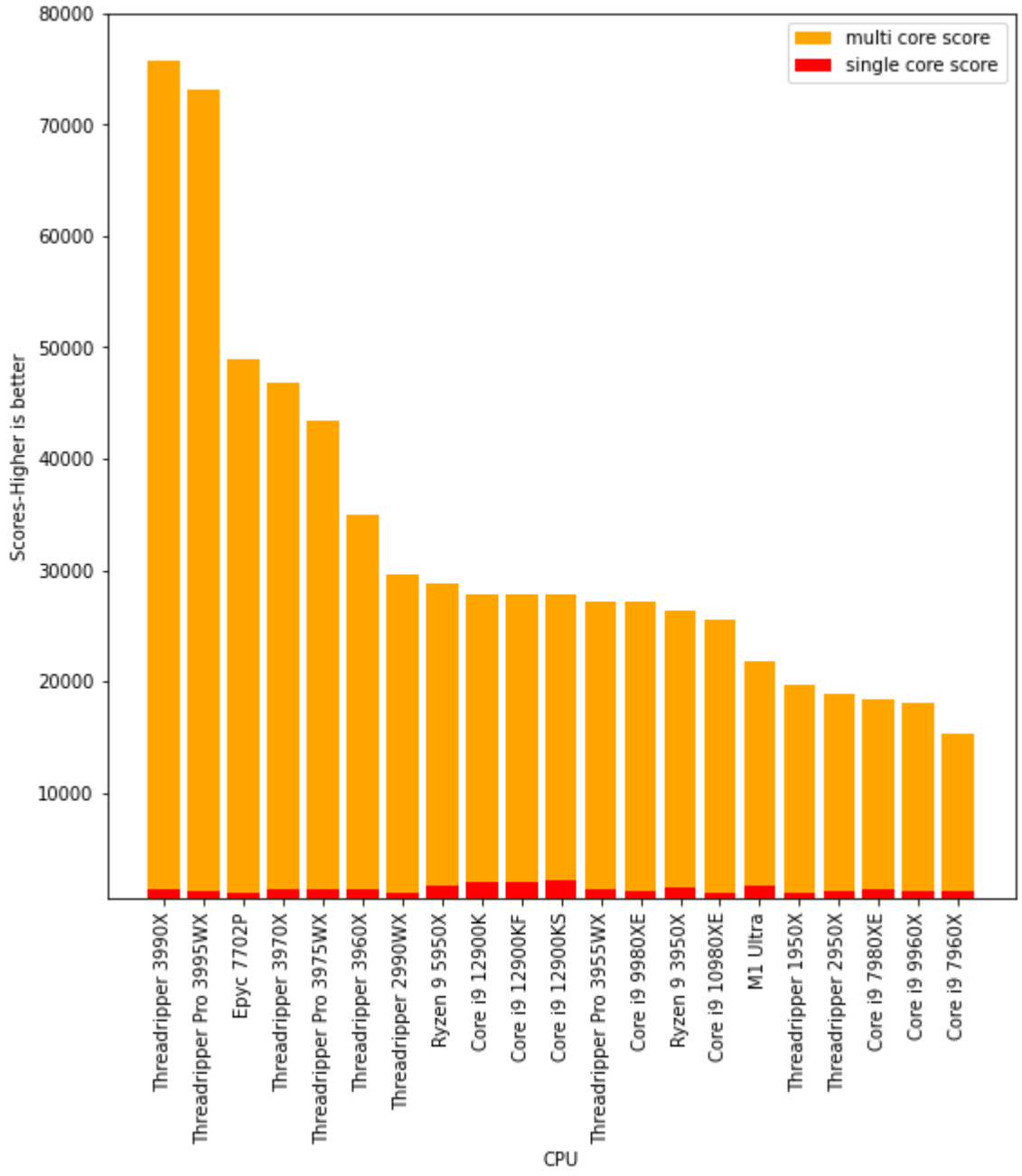
In [107]:
```
#2 What is the market share of each CPU types?
plt.figure(figsize=(8,8))
labels=['Desktop','Laptop']
cols=['gold','lightgreen']
plt.pie(a,labels=labels,colors=cols,autopct='%1.1f%%')
plt.title('Dekstop vs Laptop Market Share')
plt.legend()
plt.show()
```



In [58]:
```
#3 Which CPU have 16 cores and above?
b=df[df['cores']>=16]
plt.figure(figsize=(9,9))
x,y=b['cpuName'],b['cores']
plt.xticks(rotation=90)
plt.xlabel('CPU')
plt.ylabel('No of cores')
plt.bar(x,y,color='r')
plt.show()
```

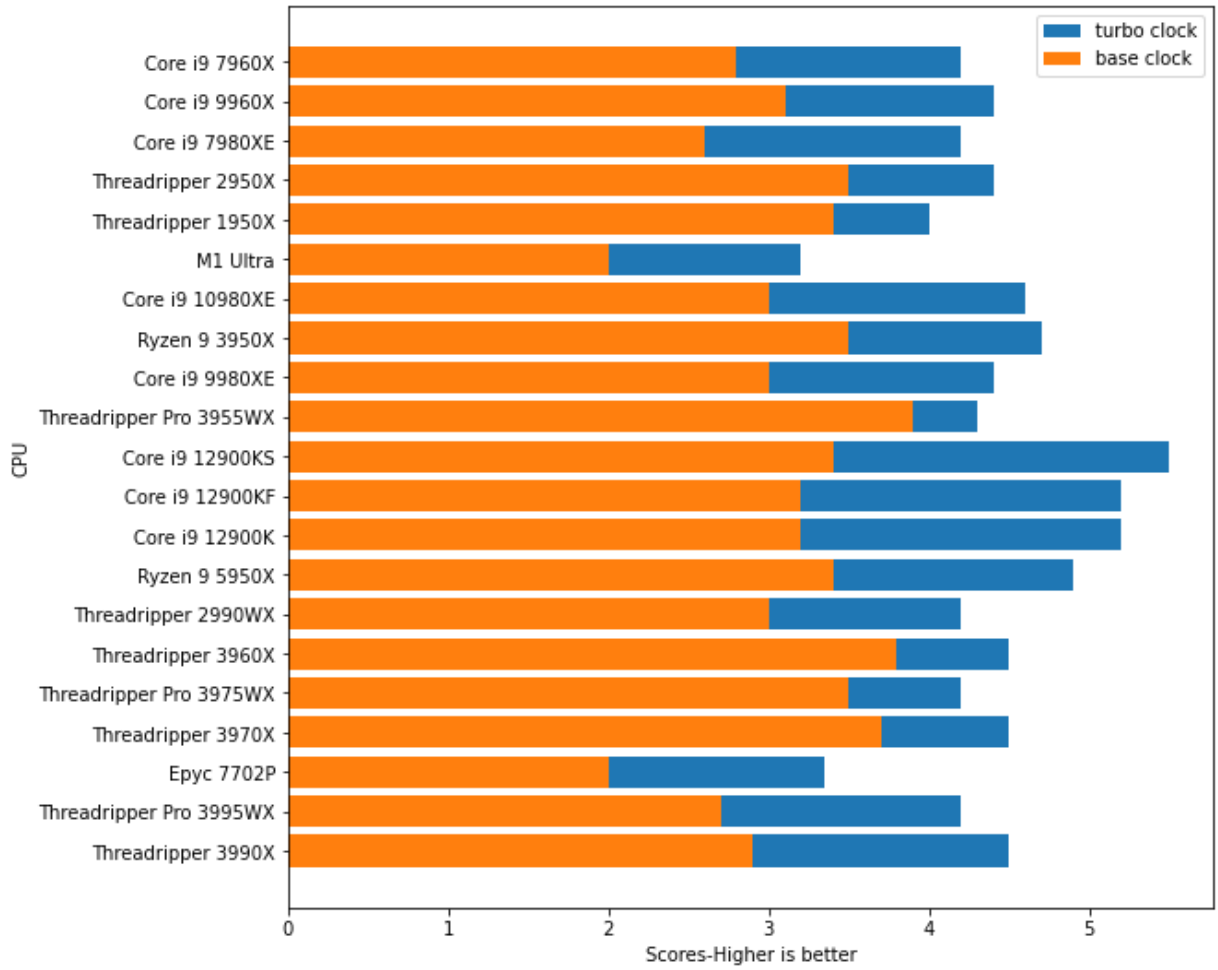


In [143]:
```
#4 What is the total benchamrk scores of top 16 processors?
b=df[df['cores']>=16]
plt.figure(figsize=(9,9))
x,y=b['cpuName'],b['singleScore']
x,y1=b['cpuName'],b['multiScore']
plt.bar(x,y1,color='orange',label='multi core score')
plt.bar(x,y,color='red',label='single core score')
plt.xticks(rotation=90)
plt.ylim(500,80000)
plt.xlabel('CPU')
plt.ylabel('Scores-Higher is better')
plt.legend()
plt.show()
```



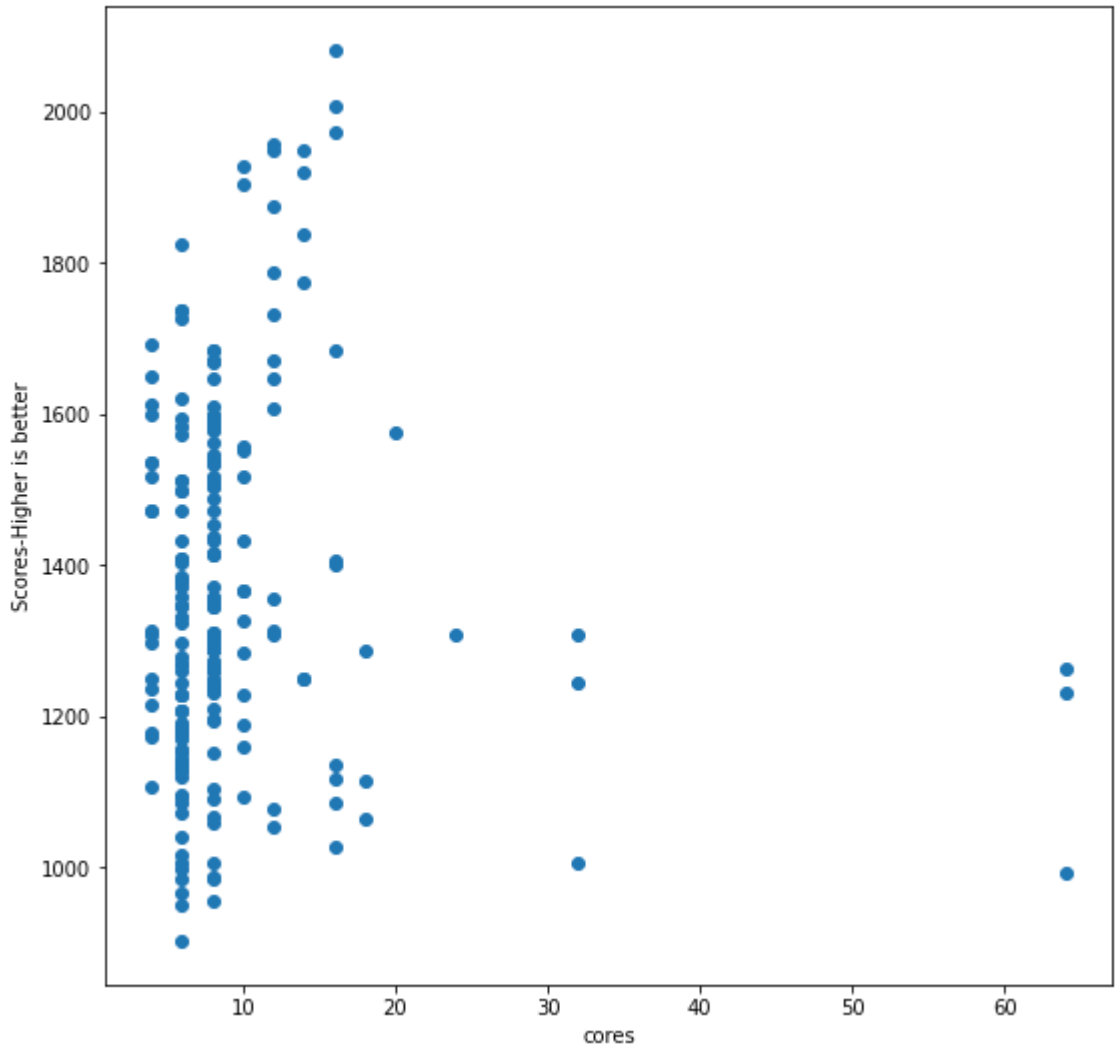
The above plot shows that AMD's threadripper series processors has the overall highest benchmark scores while Intel processors has lowest overall benchmark scores from the top 16 processors

In [117]:
```
#5 What is the base clock and turbo clock of top 16 processors?
b=df[df['cores']>=16]
plt.figure(figsize=(9,9))
x,y=b['cpuName'],b['baseClock']
x,y1=b['cpuName'],b['turboClock']
plt.barh(x,y1,label='turbo clock')
plt.barh(x,y,label='base clock')
plt.ylabel('CPU')
plt.xlabel('Scores-Higher is better')
plt.legend()
plt.show()
```



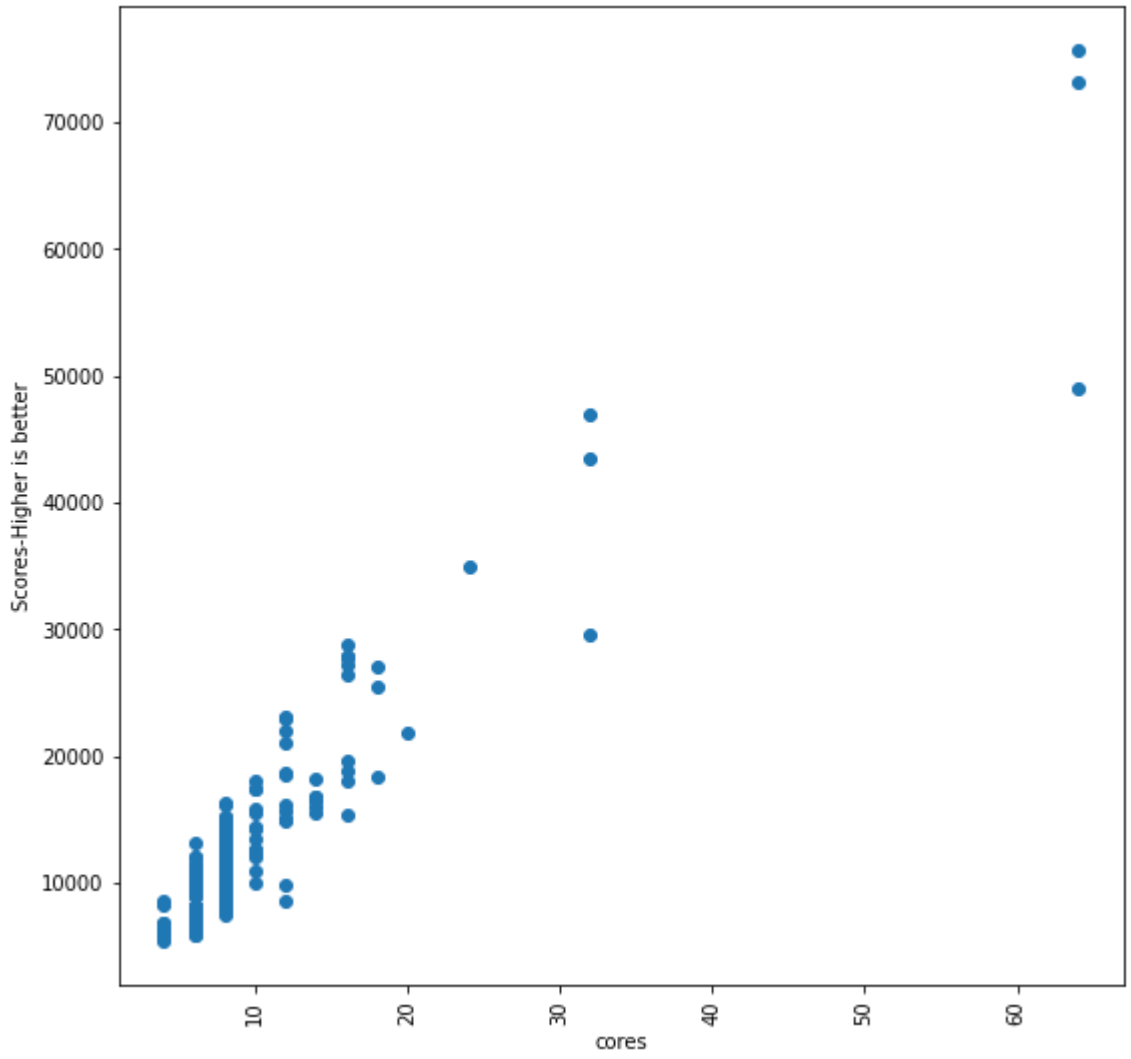
The above bar graph shows that intel has the highest turbo clock of about 5.5 ghz while AMD's threadripper has the highest base clock of about 4 ghz. So one can choose the better processor depending on the applications

In [116]:
```
#6 Cores vs single core performance
plt.figure(figsize=(9,9))
x,y=df['cores'],df['singleScore']
plt.scatter(x,y)
plt.xlabel('cores')
plt.ylabel('Scores-Higher is better')
plt.show()
```



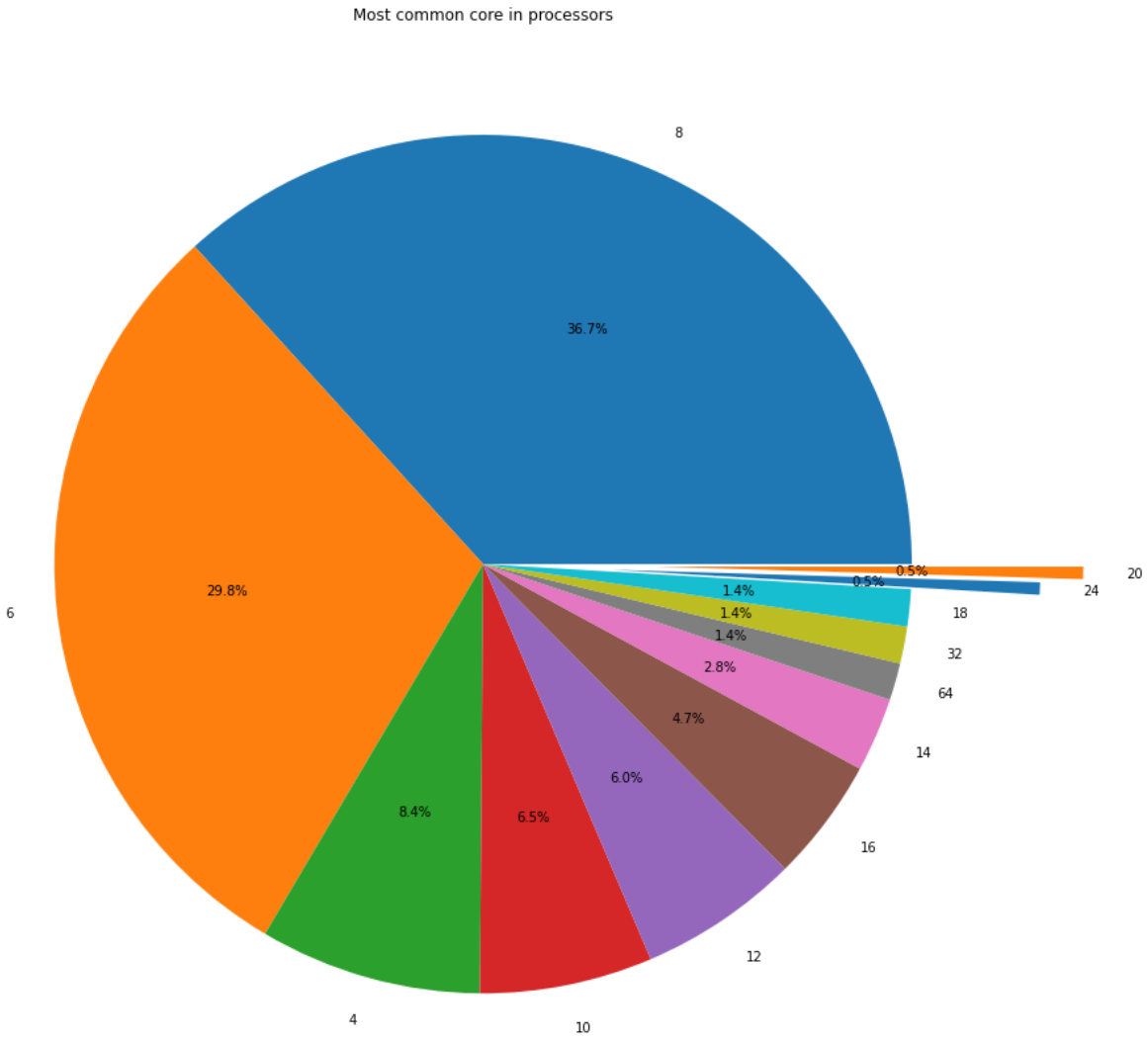
Insight- The above scatter plot shows us that the cores with lower number of cores has better single threaded performance as compared to processors with more cores

In [100]:
```
# Cores vs Multi core performance
plt.figure(figsize=(9,9))
x,y=df['cores'],df['multiScore']
plt.scatter(x,y)
plt.xticks(rotation=90)
plt.xlabel('cores')
plt.ylabel('Scores-Higher is better')
plt.show()
```



Insight- The above scatter plot shows us that the cores with higher number of cores has better multi threaded performance as compared to processors with lower cores

In [156]:
```
#8 Which is the most common core used by the processors?
a=df['cores'].value_counts()
plt.figure(figsize=(15,15))
plt.pie(a,labels=[8,6,4,10,12,16,14,64,32,18,24,20],autopct="%1.1f%%",
        explode=(0,0,0,0,0,0,0,0,0,0,0.3,0.4))
plt.title('Most common core in processors')
plt.show()
```



Above graph plot shows that the majority of the processors in the market use 8 cores as its base and is considered to be the most preferred type of core of processors.

In [155]:
```
#9.processors to avoid while buying from the list? i.e least single and multi sco
lslm=df[df['singleScore']==df['singleScore'].min()]
lslm[lslm['multiScore']==lslm['multiScore'].min()]
lslm
```

Out[155]:

	manufacturer	cpuName	singleScore	multiScore	cores	threads	baseClock	turboClock	
198	AMD	Ryzen 5 1600	903	6282	6	12	3.2	3.6	Des

In [152]:
```
#10 Least singleScore processor?
lsp=df.groupby(['singleScore'])
lsp.get_group(df['singleScore'].min())
```

Out[152]:

	manufacturer	cpuName	singleScore	multiScore	cores	threads	baseClock	turboClock	
198	AMD	Ryzen 5 1600	903	6282	6	12	3.2	3.6	Des

In [154]:
```
#10 Least multiScore processor?
lmp=df.groupby(['multiScore'])
lmp.get_group(df['multiScore'].min())
```

Out[154]:

	manufacturer	cpuName	singleScore	multiScore	cores	threads	baseClock	turboClock	
214	AMD	Ryzen 3 3100	1105	5423	4	8	3.6	3.9	Des

In [ ]:
```

```