

به نام خدا

تمرین دوم
رایانش سبز

سپند حقیقی

۹۵۲۱۰۰۷۹

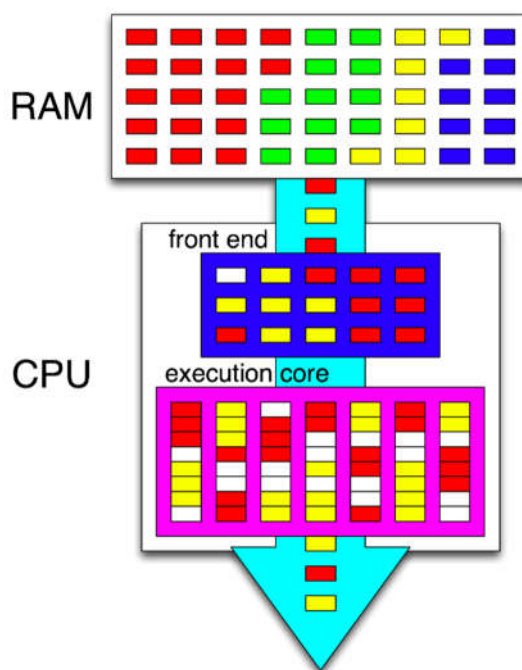
فروردین – ۱۳۹۶

۱-

الف)

فراریسمانی یا Hyper-Threading نام فناوری چندریسمانی همزمان پیاده سازی شده در پردازنده های جدید شرکت اینتل است. فراریسمانی یک فناوری اختصاصی شرکت اینتل است که برای بهبود رایانش موازی در پردازنده به کار گرفته می شود. با این فناوری برای هر هسته پردازشی (واقعی) سیستم عامل به دو هسته مجازی آدرس دهی می کند و در هنگامی که امکان داشته باشد حجم کار را بین آنها تقسیم می کند. برای این فناوری نه تنها لازم است که سیستم عامل از چندپردازه پشتیبانی کند بلکه باید برای آن بهینه سازی شده باشد.

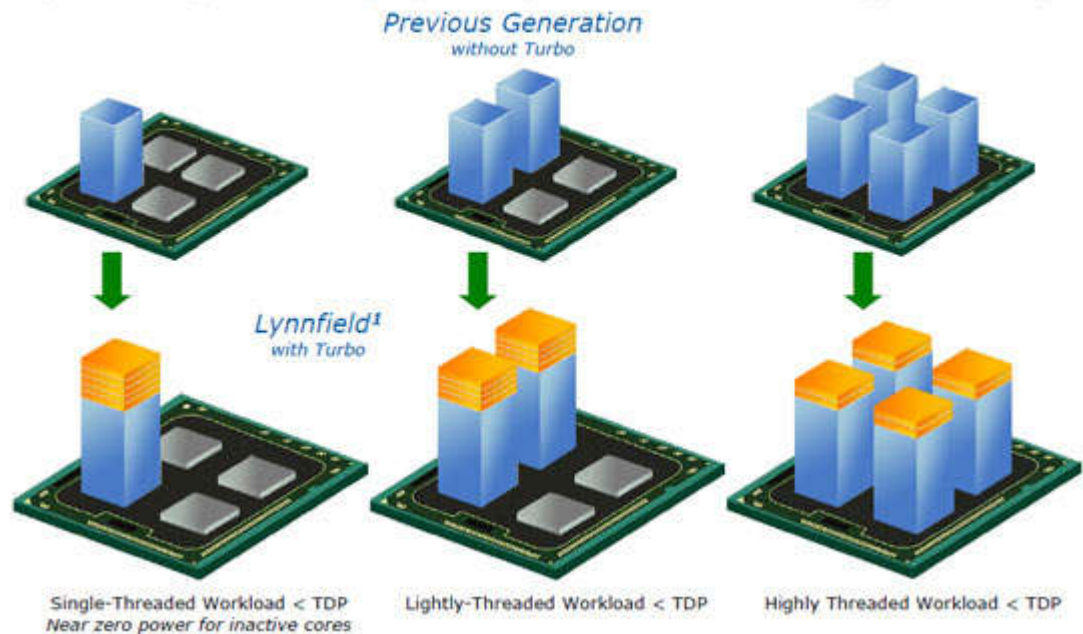
هایپرتدینگ به وسیله دوبر کردن قسمت های از پردازنده (قسمت هایی حالت یک پروسه را ذخیره می کنند) عمل می کند. به این ترتیب یک پردازنده با این فناوری می تواند به صورت دو پردازنده منطقی به سیستم عامل میزبان شناسانده شود و سیستم عامل به طور همزمان دو رشته عملیاتی را برای آنها زمان بندی کند. هنگامی که منابع پردازشی توسط یک پردازنده استفاده نمی شود و مخصوصا هنگامی که پردازنده (به دلایل مختلف) معلق شده است، یک پردازنده مجهز به این فناوری قادر است از این منابع پردازشی برای اجرای پروسه های زمان بندی شده دیگر استفاده کند. نمای کلی این فناوری در شکل ۱- آمده است.^[1]



شکل ۱- (Hyper-Threading)

توربو بوست (Turbo Boost) قابلیت در پردازنده های اینتل است که به صورت پویا سرعت پردازنده را تغییر می دهد. این افزایش سرعت در مواقع نیاز و به درخواست سیستم عامل رخ می دهد تا پردازنده بتواند در زمان بیکاری و انجام کارهای سبک تر انرژی کمتری مصرف کند.[2]

Intel® Turbo Boost Technology¹ in mainstream Dynamically delivering optimal performance & energy efficiency



شکل-۲ (Turbo Boost)

^۱ RAPL یک پلتفرم نرم افزاری به منظور مشاهده و کنترل توان مصرفی سیستم می باشد.

RAPL-Counters کل زمانی که مکانیزم P_State را زیر حالت درخواست شده سیستم عامل نگه داشته است را اندازه گیری می کند. همچنین این شمارنده ها کل زمانی که سیستم محدود کننده فرکانس فعال بوده است را ذخیره می کند. از این شمارنده ها می توان برای بالانس بار بر روی گره های مختلف سیستم استفاده کرد.[3]

¹ Running Average Power Limit

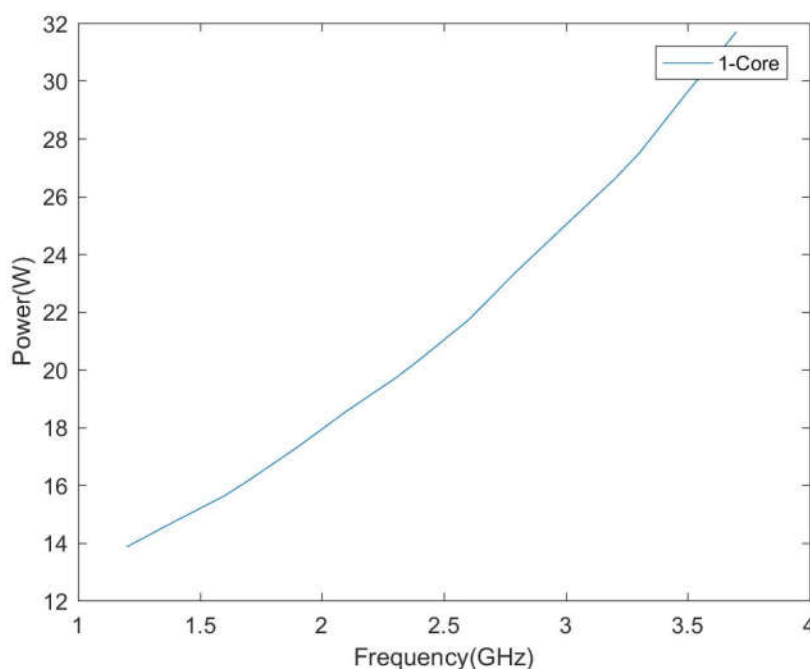
چند ریسمانی (Multi Threading) به معنای توانایی تقسیم یک پردازش به بخش های کوچکتر و انجام آن به صورت همزمان است. در این صورت سرعت اجرای فرایند افزایش می یابد.^[4]

Hyper-Threading یک فرایند چند ریسمانی همزمان^۲ به معنای تکنیکی برای افزایش بازدهی کلی پردازنده هایی که بیش از یک دستورالعمل را همزمان اجرا می کنند همراه با چند ریسمانی سخت افزاری است. این فناوری به ریسه های مستقل از نظر سخت افزاری اجزا می دهد که از منابعی که در طراحی پردازنده در اختیار آن ها قرار داده شده است، بهتر استفاده کنند.

برای فعال سازی هریک از این مکانیزم ها باید رجیستر مربوط به آن ها مقدار دهی شود.

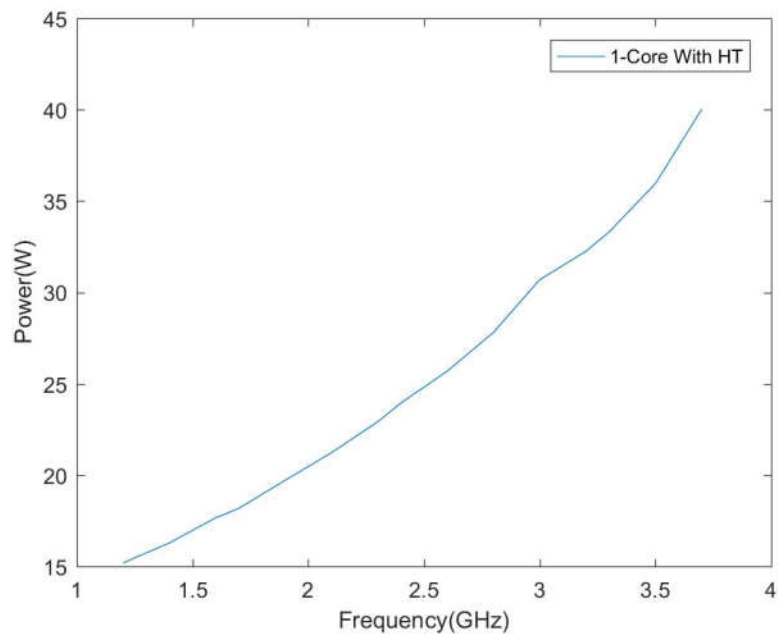
(ب)

نمودار های مصرف توان برای ۳ حالت تک هسته ای، دو هسته ای و تک هسته ای با تکنولوژی Hyper-Threading در شکل های ۳ الی ۵ آمده است.

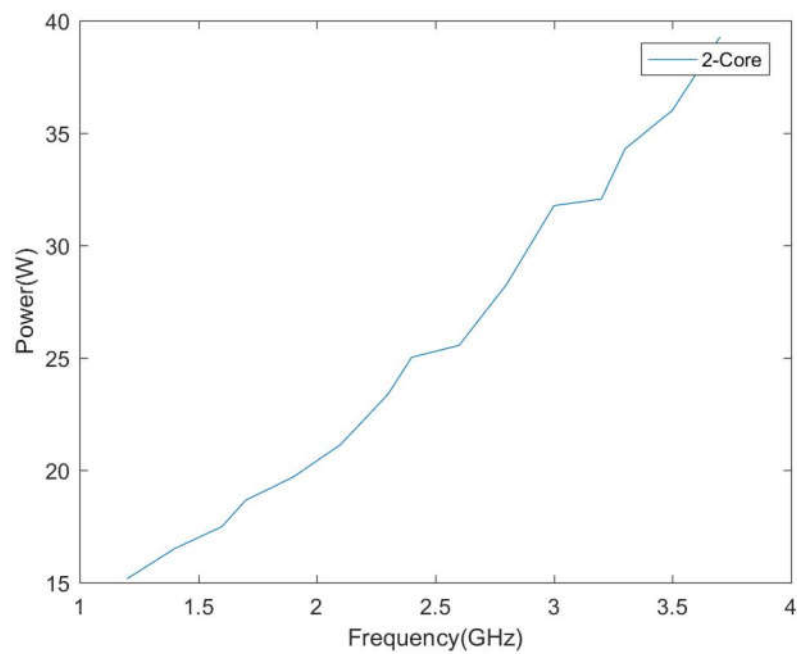


شکل-۳ (مصرف توان بر حسب فرکانس تک هسته ای)

² Simultaneous multithreading –SMT



شکل-۴ (مصرف توان بر حسب فرکانس تک هسته ای با تکنولوژی HT)



شکل-۵ (مصرف توان بر حسب فرکانس دو هسته ای)

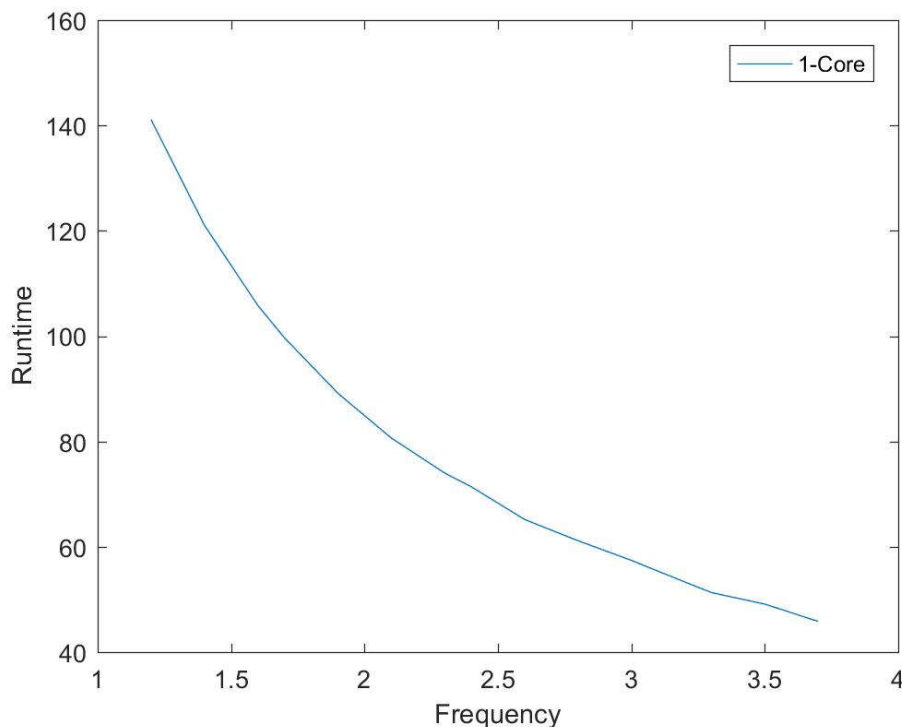
همان طور که انتظار می رود با افزایش فرکانس میزان توان مصرفی در هر سه حالت اندازه گیری شده افزایش می یابد.

میزان مصرف در حالت تک هسته ای از دو حالت دیگر به شکل قابل ملاحظه ای کمتر است که دلیل آن عدم وجود فعالیت اضافه (سخت افزاری و نرم افزاری) برای موازی سازی می باشد.

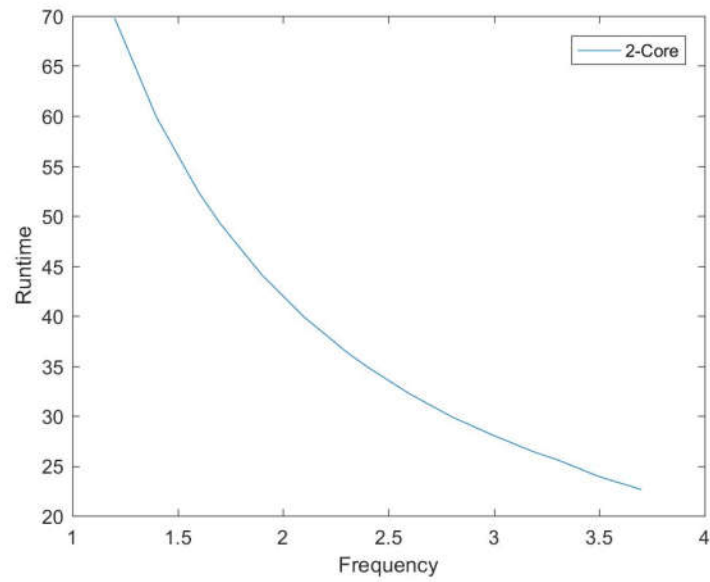
میزان مصرف توان در حالت تک هسته ای با تکنولوژی Hyper-Threading از مصرف در حالت دو هسته ای بیشتر است که علت آن صرف انرژی زیادتر برای موازی سازی در حالت استفاده از Hyper-Threading می باشد که قسمتی از آن به علت استفاده از پردازنده های superscalar است.

ج

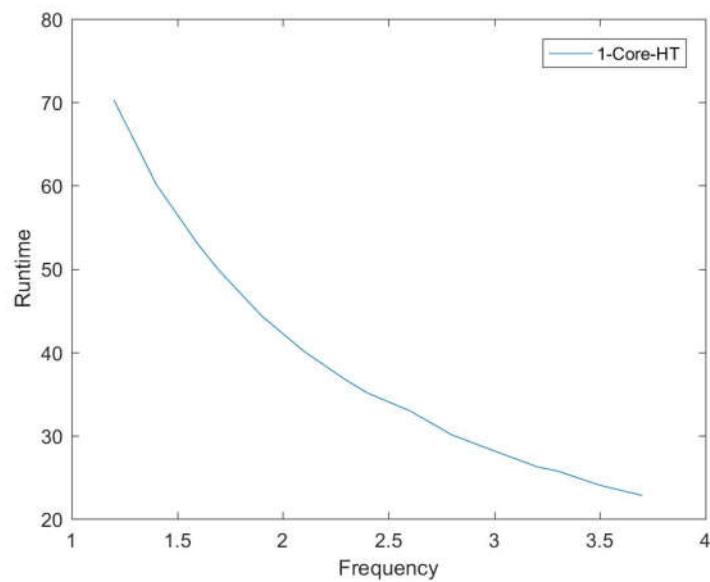
نمودارهای زمان اجرا برای ۳ حالت تک هسته ای، دو هسته ای و تک هسته ای با تکنولوژی Hyper-Threading در شکل های ۶ الی ۸ آمده است.



شکل ۶- (زمان اجرا بر حسب فرکانس تک هسته ای)



شکل-۷ (زمان اجرا بر حسب فرکانس دو هسته ای)



شکل-۸ (زمان اجرا بر حسب فرکانس تک هسته ای با تکنولوژی HT)

همان طور که انتظار داشتیم با افزایش فرکانس کاری پردازنده زمان اجرا سیستم برای هر ۳ حالت کاهش پیدا می کند. با توزیع تسک بر روی هسته های مختلف با توجه به موازی انجام شدن محاسبات مختلف اندازه زمان اجرا به صورت قابل ملاحظه ای کاهش پیدا می کند.

متوسط زمان اجرا برای حالت تک هسته ای ۷۷,۷۵۰۷ ، برای حالت دو هسته ای ۳۸,۳۳۹۸ و برای حالت تک هسته با استفاده از تکنولوژی Hyper-Threading ۳۸,۶۲۰۷ واحد زمانی می باشد. که متوسط درصد کاهش زمان اجرا برای دو حالت دو هسته ای و تک هسته ای با تکنولوژی HT به صورت زیر است.

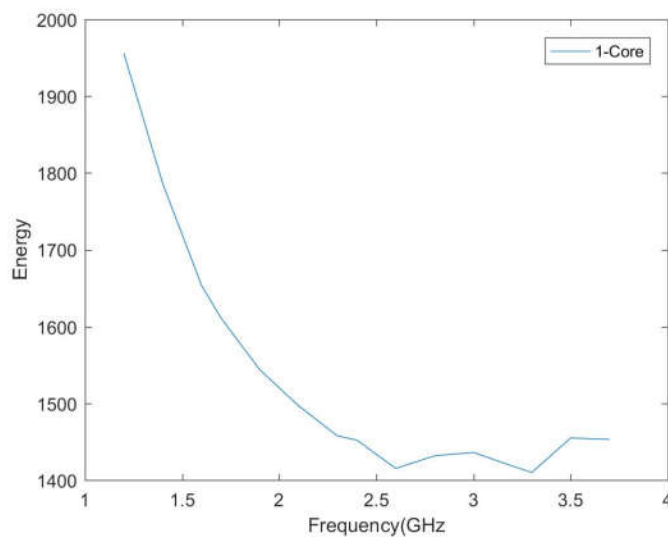
$$2 - core \rightarrow \frac{39.4109}{77.7507} = 50.6\%$$

$$1 - core(HT) \rightarrow \frac{39.13}{77.7507} = 50.3\%$$

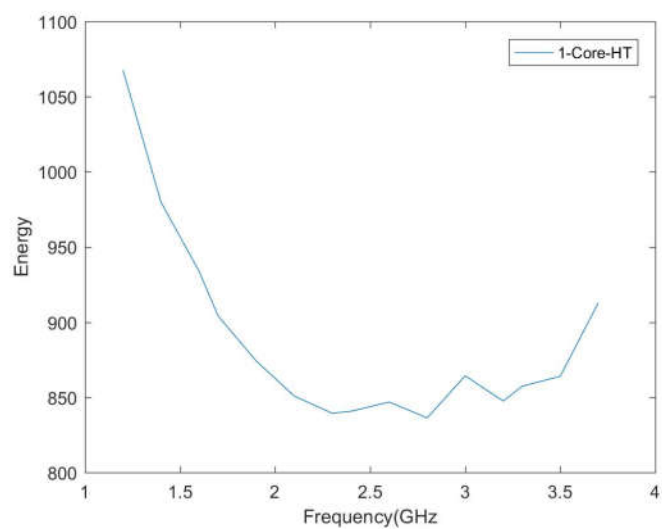
همان طور که مشاهده می شود میزان کاهش زمان اجرا در حالت ۲ هسته ای مقدار اندکی از حالت استفاده از تکنولوژی HT بیشتر است که علت آن استفاده از دو پردازنده واقعی به جای دو پردازنده مجازی می باشد.

(د)

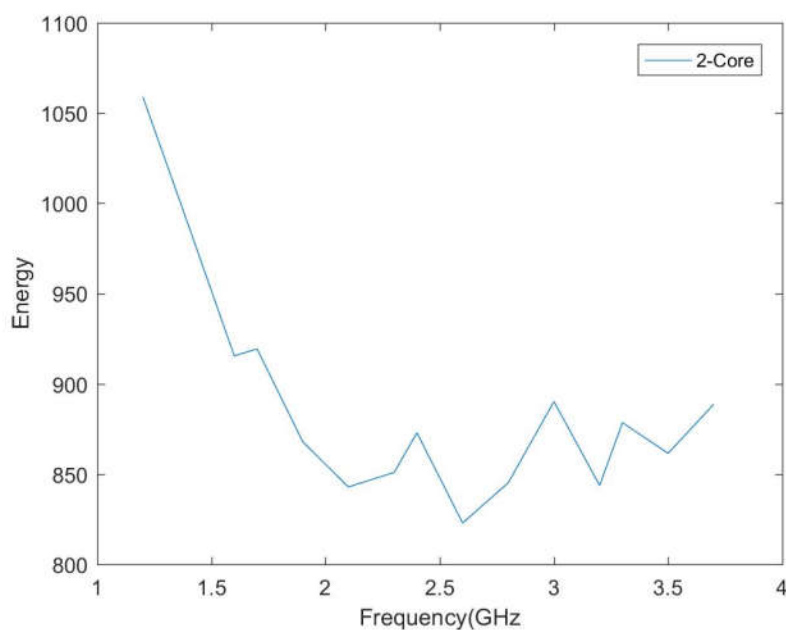
نمودار های میزان مصرف انرژی برای ۳ حالت تک هسته ای، دو هسته ای و تک هسته ای با تکنولوژی Hyper-Threading در شکل های ۹ الی ۱۱ آمده است.



شکل-۹ (میزان مصرف انرژی بر حسب فرکانس تک هسته ای)



شکل-۱۰ (میزان مصرف انرژی بر حسب فرکانس تک هسته ای با تکنولوژی HT)



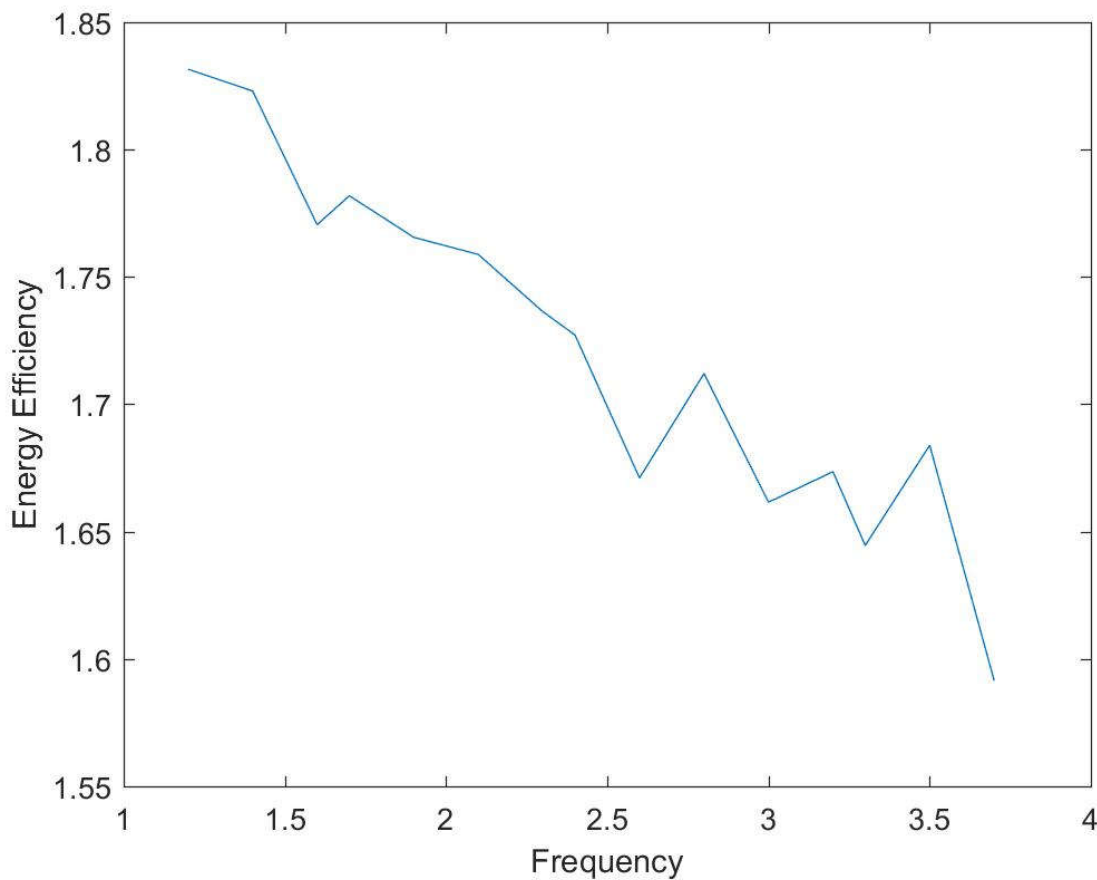
شکل-۱۱ (میزان مصرف انرژی بر حسب فرکانس دو هسته ای)

همان طور که در شکل ها مشاهده می شود میزان مصرف انرژی در ابتدا با افزایش فرکانس به صورت تقریباً خطی کاهش پیدا می کند ولی از یک فرکانس به بعد این میزان متوقف شده و افزایش پیدا می کند.

انرژی مصرفی به صورت حاصلضرب توان مصرفی در زمان اجرا بیان می شود و علت این افزایش انرژی برابر نبودن شیب رشد توان مصرفی پردازنده و کاهش زمان اجرا می باشد.

(۵)

نمودار بازدهی انرژی در شکل-۱۲ آمده است.



شکل-۱۲ (نمودار بازدهی انرژی بر حسب فرکانس دو هسته ای)

همان طور که در نمودار مشخص است بازدهی انرژی با افزایش فرکانس کاهش پیدا کرده و به مرور به ۱ میل می کند که علت برقرار شدن یک تعادل بین توان مصرفی بیشتر به علت افزایش فرکانس در حالت چند هسته ای و کاهش زمان اجرا در این حالت می باشد. این به این معنا است که در فرکانس های بالا مصرف انرژی سیستم تک هسته ای و یک سیستم اجرای موازی در یک بازه هستند.

۲-

(الف)

توان مصرفی هر سیستم با فرض حداکثر مصرف هر قطعه به صورت زیر محاسبه می شود :

$$P_{Sys1} = 79W + 3.7W + 7W = 89.7W$$

$$P_{Sys2} = 66W + 7.9W + 2.3W = 76.2W$$

میزان مصرف توان منبع تغذیه با توجه به بازده ۷۰ درصدی برای هر سیستم به صورت زیر محاسبه می شود :

$$P_{S-Sys1} = \frac{89.7W \times 10}{7} = 128.14W$$

$$P_{S-Sys2} = \frac{76.2W \times 10}{7} = 108.85W$$

(ب)

با فرض اینکه ۴۰ درصد از زمان هارد-۱ در حالت idle است مقدار متوسط توان مصرفی آن به صورت زیر است :

$$P_{H-1} = 0.4 \times 2.9W + 0.6 \times 7.0W = 5.36W$$

(ج)


برای اینکه هارد-۲ به صورت متوسط ۵,۶ وات توان مصرف کند باید تساوی زیر برقرار باشد :

$$P_{H-2} = t_{idle} \times 4.0W + 1 - t_{idle} \times 7.9W = 5.36W \rightarrow t_{idle} = 0.65$$

با این شرایط اگر این هارد ۶۵ درصد مواقع در حالت idle باشد به اندازه هارد قبلی توان مصرف می کند.

(د)

برای این قسمت برنامه ای به زبان برنامه نویسی پایتون نوشته شده است که اطلاعات میزان مصرف پردازنده ، هارد و رم را به صورت فایل های رفرنس (با فرمت .ref) و همچنین درصد زمان اشغال بودن هر یک از قطعات را از کاربر دریافت می کند و در خروجی از بین قطعات موجود در فایل های رفرنس بهترین پیکره بندی از نظر مصرف توان را نشان می دهد. (کد برنامه در پیوست موجود است)^۳



```
C:\Users\Sepkjaer\AppData\Local\Programs\Python\Python35-32
Please Enter RAM Usage Percent [0,100] 20
Please Enter HARD Usage Percent [0,100] 50
Please Enter CPU Usage Percent [0,100] 30
RAM :RAM-2
CPU :CPU-2
HARD :Hard-1
Total :68.11999999999999 W

Process finished with exit code 0
```

شکل-۱۳ (خروجی نرم افزار نوشته شده)

³ <https://github.com/sepandhaghighi/Green-HW/tree/master/HW-2/App>

در این قسمت از نرم افزار **linpack** برای ارزیابی قدرت محاسباتی سیستم استفاده شده است. فایل تنظیمات این برنامه در شکل-۱۴ آمده است.

```

1 Sample Intel(R) Optimized LINPACK Benchmark data file (lininput_xeon64)
2 Intel(R) Optimized LINPACK Benchmark data
3 1 # number of tests
4 30000 30000 # problem sizes
5 30000 30000 # leading dimensions
6 1 2 # times to run a test
7 4 4 # alignment values (in KBytes)
8

```

شکل-۱۴ (تنظیمات برنامه linpack)

ابعاد مساله ۳۰۰۰۰ در ۳۰۰۰۰ در نظر گرفته شده است.

پس از اتمام اجرای برنامه فایل خروجی برنامه تولید و میزان محاسبات را نشان می دهد. این فایل خروجی در شکل-۱۵ آمده است.

```

1 Intel(R) Optimized LINPACK Benchmark data
2
3 Current date/time: Sat Apr 15 12:19:19 2017
4
5 CPU frequency: 3.253 GHz
6 Number of CPUs: 1
7 Number of cores: 4
8 Number of threads: 4
9
10 Parameters are set to:
11
12 Number of tests: 1
13
14 Number of equations to solve (problem size) : 30000
15 Leading dimension of array : 30000
16 Number of trials to run : 1
17 Data alignment value (in Kbytes) : 4
18 Maximum memory requested that can be used=7200604096, at the size=30000
19
20 ===== Timing linear equation system solver =====
21
22 Size LDA Align. Time(s) GFlops Residual Residual(norm) Check
23 30000 30000 4 134.775 133.5690 8.725493e-10 3.439598e-02 pass
24
25 Performance Summary (GFlops)
26
27 Size LDA Align. Average Maximal
28 30000 30000 4 133.5690 133.5690
29
30 Residual checks PASSED

```

شکل-۱۵ (خروجی برنامه linpack)

برای اندازه گیری توان مصرفی سیستم در سیستم عامل ویندوز از برنامه Joulemeter در حالت کاری باتری استفاده شده است.

خروجی این برنامه در جدول ۱- آمده است.

	A	B	C	D	E	F	G
1	TimeStamp	Total Power	CPU (W)	Monitor (W)	Disk (W)	Base (W)	
2	6.36E+13	23.2	0.2	8	0	15	
3	6.36E+13	24.2	1.2	8	0	15	
4	6.36E+13	24.7	1.7	8	0	15	
5	6.36E+13	37.3	14.3	8	0	15	
6	6.36E+13	37.3	14.3	8	0	15	
7	6.36E+13	37.5	14.5	8	0	15	
8	6.36E+13	37	14	8	0	15	
9	6.36E+13	37	14	8	0	15	
10	6.36E+13	36.9	13.9	8	0	15	
11	6.36E+13	37.1	14	8	0	15	
12	6.36E+13	38	15	8	0	15	
13	6.36E+13	37.1	14	8	0	15	
14	6.36E+13	37.2	14.2	8	0	15	
15	6.36E+13	37.3	14.3	8	0	15	
16	6.36E+13	37.1	14.1	8	0	15	
17	6.36E+13	37.2	14.2	8	0	15	
18	6.36E+13	37.2	14.2	8	0	15	
19	6.36E+13	37.1	14.1	8	0	15	
20	6.36E+13	37.1	14.1	8	0	15	
21	6.36E+13	37.2	14.2	8	0	15	
22	6.36E+13	37.3	14.3	8	0	15	
23	6.36E+13	37.1	14.1	8	0	15	
24	6.36E+13	37.2	14.2	8	0	15	
25	6.36E+13	38.1	15.1	8	0	15	
26	6.36E+13	37.3	14.3	8	0	15	
27	6.36E+13	37.3	14.3	8	0	15	
28	6.36E+13	37.5	14.5	8	0	15	
29	6.36E+13	37.1	14.1	8	0	15	
30	6.36E+13	37.4	14.4	8	0	15	
31	6.36E+13	37.1	14.1	8	0	15	
32	6.36E+13	37.1	14.1	8	0	15	
33	6.36E+13	37	14	8	0	15	
34	6.36E+13	37.1	14.1	8	0	15	
35	6.36E+13	37	14	8	0	15	

جدول ۱- (خروجی برنامه Joulemeter)

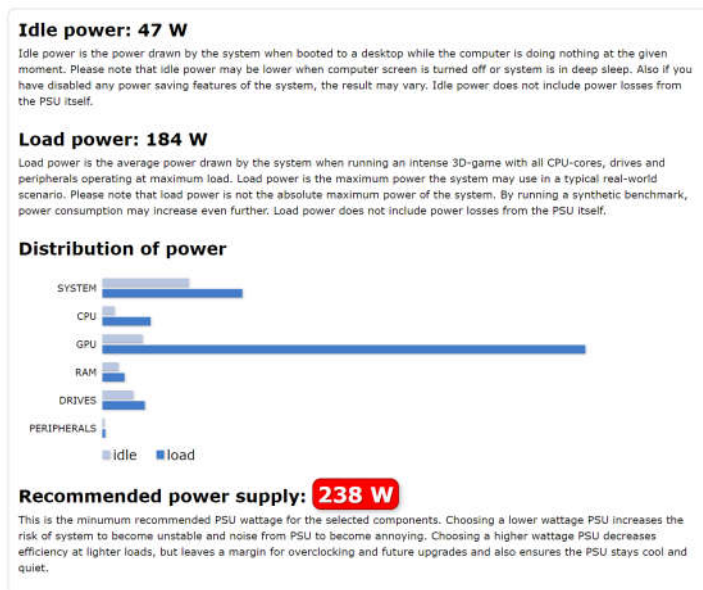
همان طور که در جدول-۱ مشاهده می شود ۳ سطر اول مربوط به عملکرد سیستم قبل از اجرای برنامه ارزیابی و سطر های بعدی مربوط به مصرف توان در زمان محاسبات است.

میانگین عملیات های ممیز شناور در این تست برابر 133.5690 GFlops اندازه گیری شده است. برای اندازه گیری میانگین عملیات ممیز شناور به ازای هر وات داریم :

$$\rightarrow \frac{Flops}{P_{system}} = \frac{133.5690 GFlops}{37W} = 3.6 GFlops - per - watt$$

در لیست ارایه شده در سایت top500 از نظر معیار میانگین عملیات های ممیز شناور به ازای هر وات بهترین سیستم دارای عملکرد 8.6Flops/w است اما بقیه لیست در بازه 2.2-4 قرار دارند که با توجه به ابررایانه بودن و تعداد هسته بسیار بالا قابل قبول است. و به طور کلی عملکرد لپ تاپ ها با توجه به مصرف بسیار پایین در این معیار بهتر است.

یکی از روش های دیگر برای اندازه گیری مصرف سیستم استفاده از ماشین حساب های اندازه گیری توان^۴ می باشد در این روش با وارد کردن مشخصات قطعات سیستم و با توجه به مقدار توان مصرفی این قطعات که در کارخانه اندازه گیری شده است توان مصرفی کل سیستم محاسبه می شود. گرچه این روش توانایی محاسبه مصرف به صورت آنلاین را ندارند. نمونه ای از خروجی یکی از این ماشین حساب ها در شکل-۱۶ آمده است.



شکل-۱۶ (خروجی برنامه ماشین حساب توان)

⁴ <http://powersupplycalculator.net/>

[1]- Hyper Threading, Wikipedia the Free Encyclopedia,
<https://en.wikipedia.org/wiki/Hyper-threading>

[2]- Intel Turbo Boost, Wikipedia the Free Encyclopedia,
https://en.wikipedia.org/wiki/Intel_Turbo_Boost

[3]- Srinivas Pandruvada, Running Average Power Limit
<https://01.org/blogs/2014/running-average-power-limit-%E2%80%93rapl>

[4]- Multithreading, Wikipedia the Free Encyclopedia,
[https://en.wikipedia.org/wiki/Multithreading_\(computer_architecture\)](https://en.wikipedia.org/wiki/Multithreading_(computer_architecture))

پیوست (کد برنامه سوال ۲)

```

import os
def get_input():
    try:
        ram_usage=float(input("Please Enter RAM Usage Percent [0,100]"))
        hard_usage = float(input("Please Enter HARD Usage Percent [0,100]"))
        cpu_usage = float(input("Please Enter CPU Usage Percent [0,100]"))
        return [ram_usage,hard_usage,cpu_usage]
    except Exception as e:
        print(e)

def read_ref():
    try:
        hard_list=[]
        ram_list=[]
        cpu_list=[]
        list_dir=os.listdir()
        if "hard.ref" not in list_dir:
            raise Exception("There is no ref file for hard disk")
        elif "ram.ref" not in list_dir:
            raise Exception("There is no ref file for hard ram")
        elif "cpu.ref" not in list_dir:
            raise Exception("There is no ref file for hard cpu")
        else:
            pass
        ram_file=open("ram.ref","r")
        hard_file=open("hard.ref","r")
        cpu_file=open("cpu.ref","r")
        user_input = get_input()
        ram_total=[]
        hard_total=[]
        cpu_total=[]
        for index,line in enumerate(ram_file):
            temp=line.strip().split(",")
            if len(temp)!=3:
                print("[Warning] RAM Ref File Out Of Format In Line "+str(index)+"-->"+str(line))
            else:
                ram_list.append(temp)

        ram_total.append((float(temp[1])*user_input[0]/100)+(float(temp[2])*(1-
user_input[0]/100)))
        for index,line in enumerate(hard_file):
            temp = line.strip().split(",")
            if len(temp)!=3:
                print("[Warning] HARD Ref File Out Of Format In Line "+str(index)+"-->"+str(line))
            else:
                hard_list.append(temp)
                hard_total.append((float(temp[1]) * user_input[1] / 100) +
float(temp[2]) * (1 - user_input[1] / 100))
        for index,line in enumerate(cpu_file):
            temp = line.strip().split(",")
            if len(temp)!=3:
                print("[Warning] CPU Ref File Out Of Format In Line "+str(index)+"-->"+str(line))

```

```

        else:
            cpu_list.append(temp)
            cpu_total.append((float(temp[1]) * user_input[2] / 100) +
float(temp[2]) * (1 - user_input[2] / 100))
            if len(ram_total)!=0:
                ram_min=min(ram_total)
                print("RAM :" + ram_list[ram_total.index(ram_min)][0])
            else:
                ram_min=0
                print("RAM : There is no item for RAM")
            if len(cpu_total)!=0:
                cpu_min=min(cpu_total)
                print("CPU :" + cpu_list[cpu_total.index(cpu_min)][0])
            else:
                cpu_min=0
                print("CPU : There is no item for CPU")

            if len(hard_list)!=0:
                hard_min=min(hard_total)
            else:
                hard_min=0
                print("HARD : There is no item for HARD")

            print("HARD :" + hard_list[hard_total.index(hard_min)][0])
            print("Total :"+str(cpu_min+hard_min+ram_min)+" W")

except Exception as e:
    print(e)

if __name__=="__main__":
    read_ref()

```