به نام خدا

تمرین دوم

رایانش سبز

سپند حقیقی

95210079

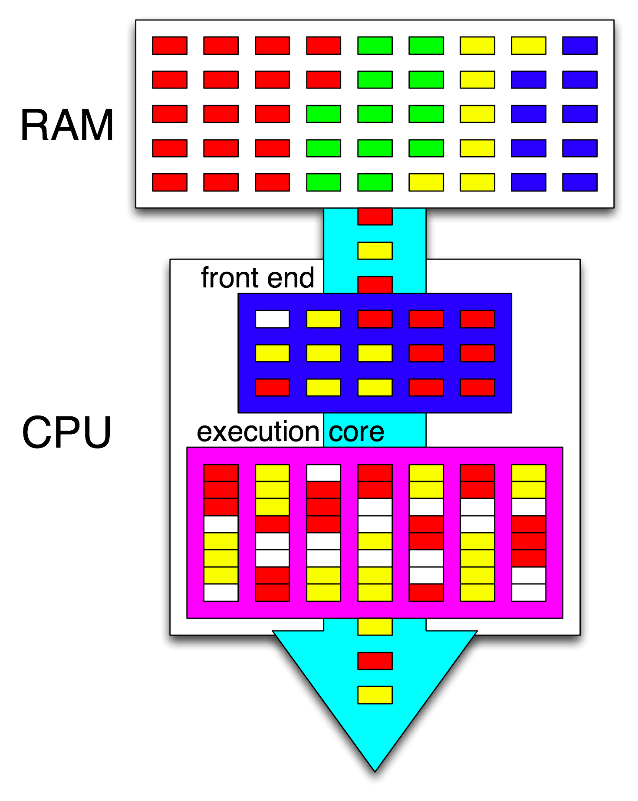
فروردین – 1396

1-

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.

هایپرتردینگ به وسیله دوبل کردن قسمت‌های از پردازنده (قسمت‌هایی حالت یک پروسه را ذخیره می‌کنند) عمل می‌کند. به این ترتیب یک پردازنده با این فناوری می‌تواند 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 نمی‌شود و مخصوصا هنگامی که پردزنده (به دلایل مختلف) معلق شده‌است، یک پردازنده مجهز به این فناوری قادر است از این منابع پردازشی برای اجرای پروسه‌های زمان بندی شده دیگر استفاده کند. نمای کلی این فناوری در شکل-1 آمده است.[1]



شکل-1 ( Hyper-Threading)

توربو بوست (Turbo Boost) قابلیتی 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]

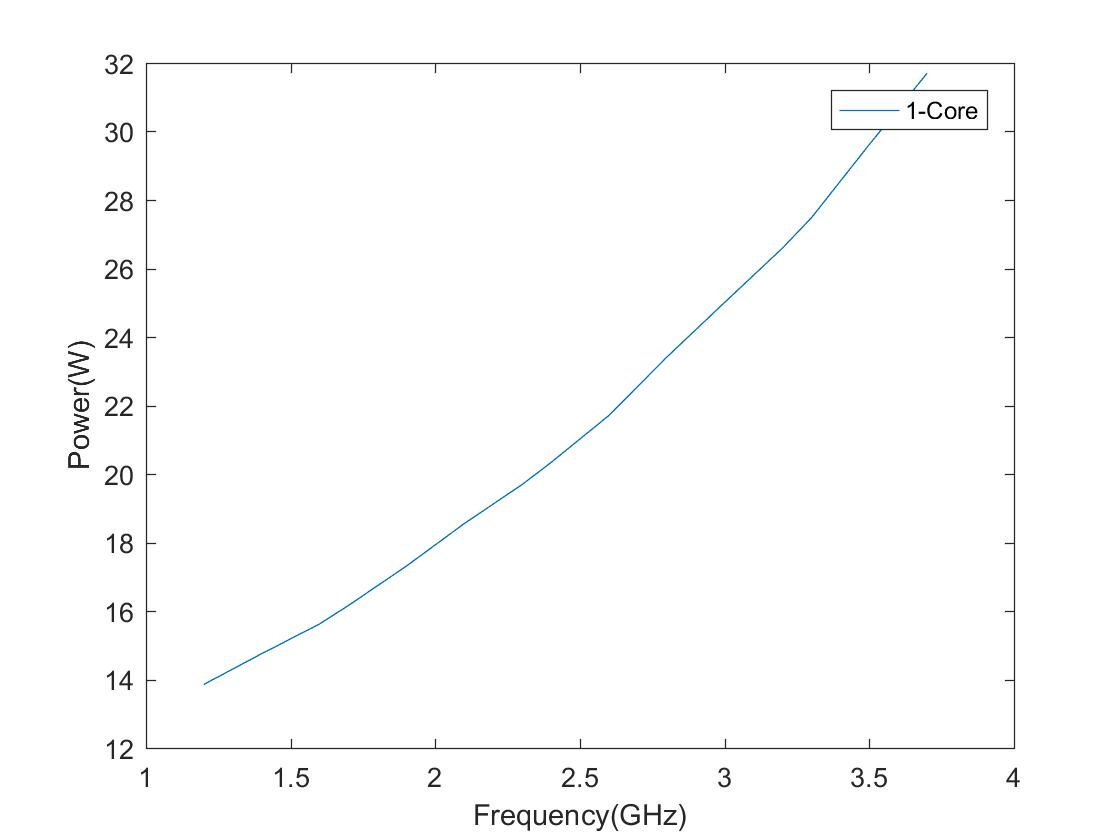
RAPL (Running Average Power Limit) یک پلتفرم نرم افزاری به منظور مشاهده و کنترل توان مصرفی سیستم می باشد. RAPL-Counters کل زمانی که مکانیزم P\_State را زیر حالت درخواست شده سیستم عامل نگه داشته است را اندازه گیری می کند. همچنین این شمارنده ها کل زمانی که سیستم محدود کننده فرکانس فعال بوده است را ذخیره می کند. از این شمارنده ها می توان برای بالانس بار برروی گره های مختلف سیستم استفاده کرد.[3]

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.

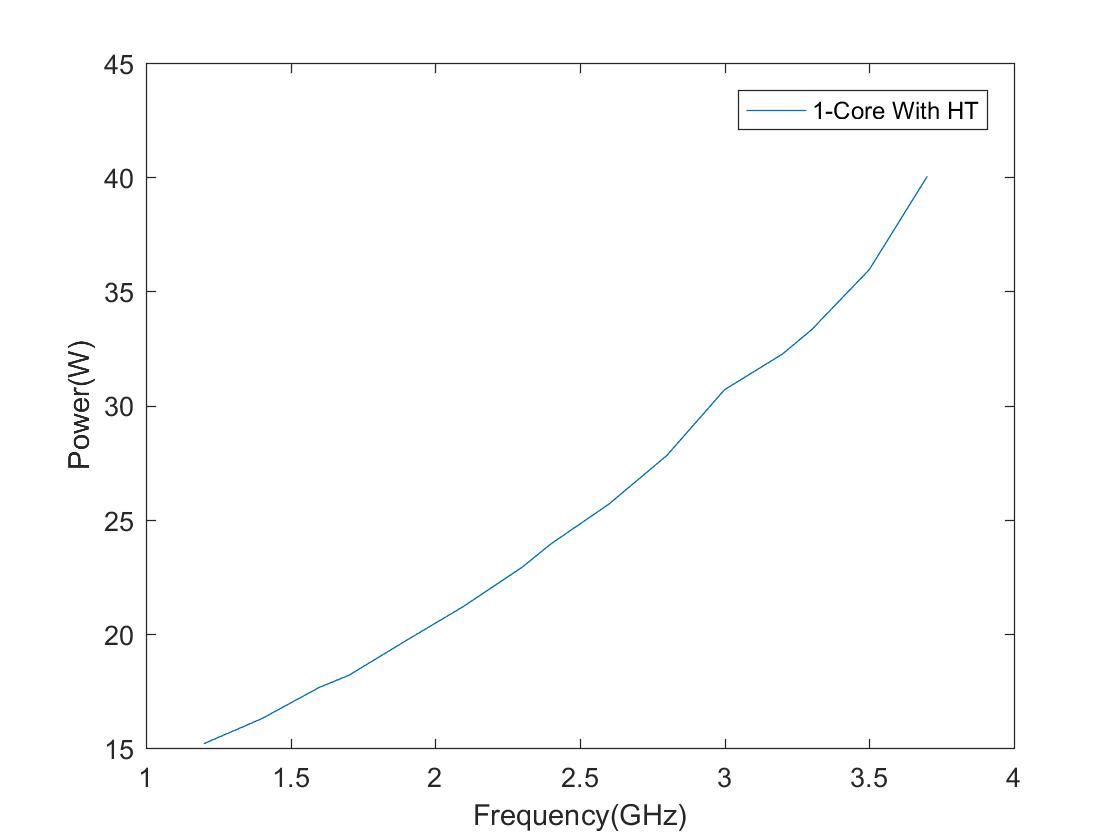
Hyper-Threading یک فرایند چند ریسمانی همزمان(Simultaneous multithreading –SMT) به معنای تکنیکی برای افزایش بازدهی کلی پردازنده هایی که بیش از یک دستورالعمل را همزمان اجرا می کنند همراه با چند ریسمانی سخت افزاری است. این فناوری به ریسه های مستقل از نظر سخت افزاری اجزا می دهد که از منابعی که در طراحی پردازنده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

ب)

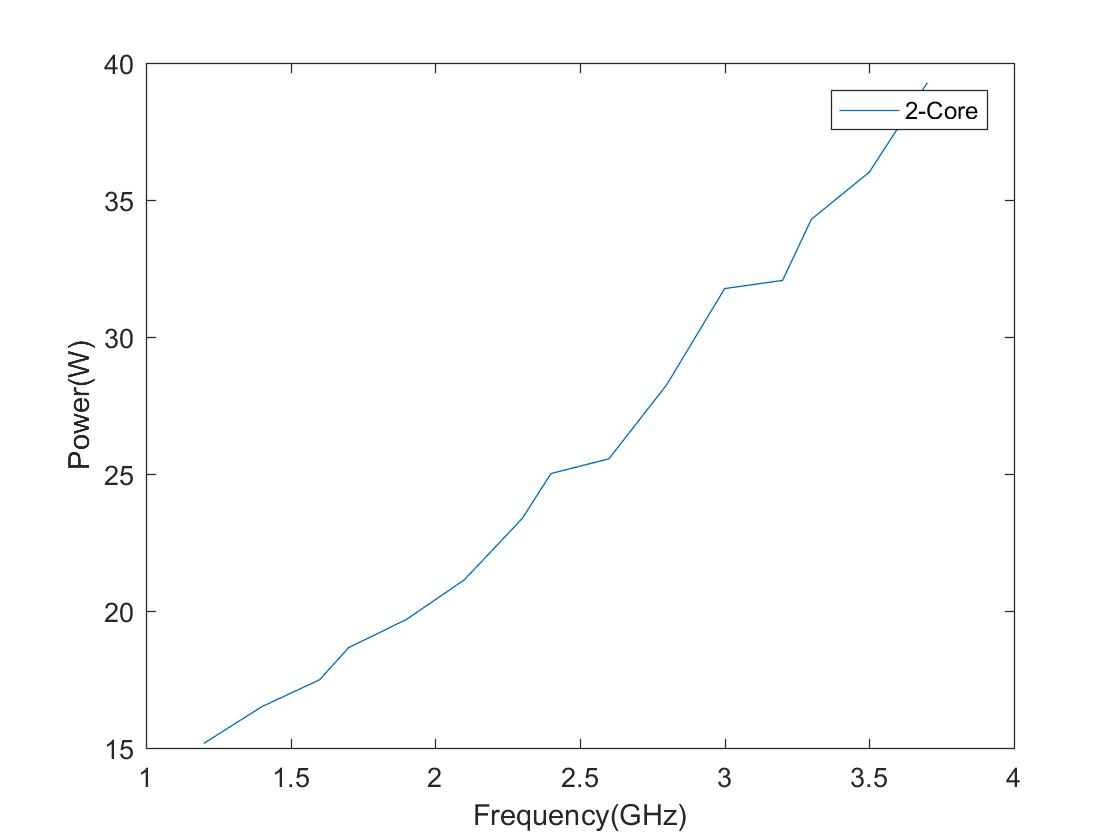
نمودار های مصرف توان برای 3 حالت تک هسته ای، دو هسته ای و تک هسته ای با تکنولوژی Hyper-Threading در شکل های 2 الی 4 آمده است.



شکل-2 ( مصرف توان بر حسب فرکانس تک هسته ای)



شکل-3 ( مصرف توان بر حسب فرکانس تک هسته ای با تکنولوژی HT )



شکل-4 ( مصرف توان بر حسب فرکانس دو هسته ای)

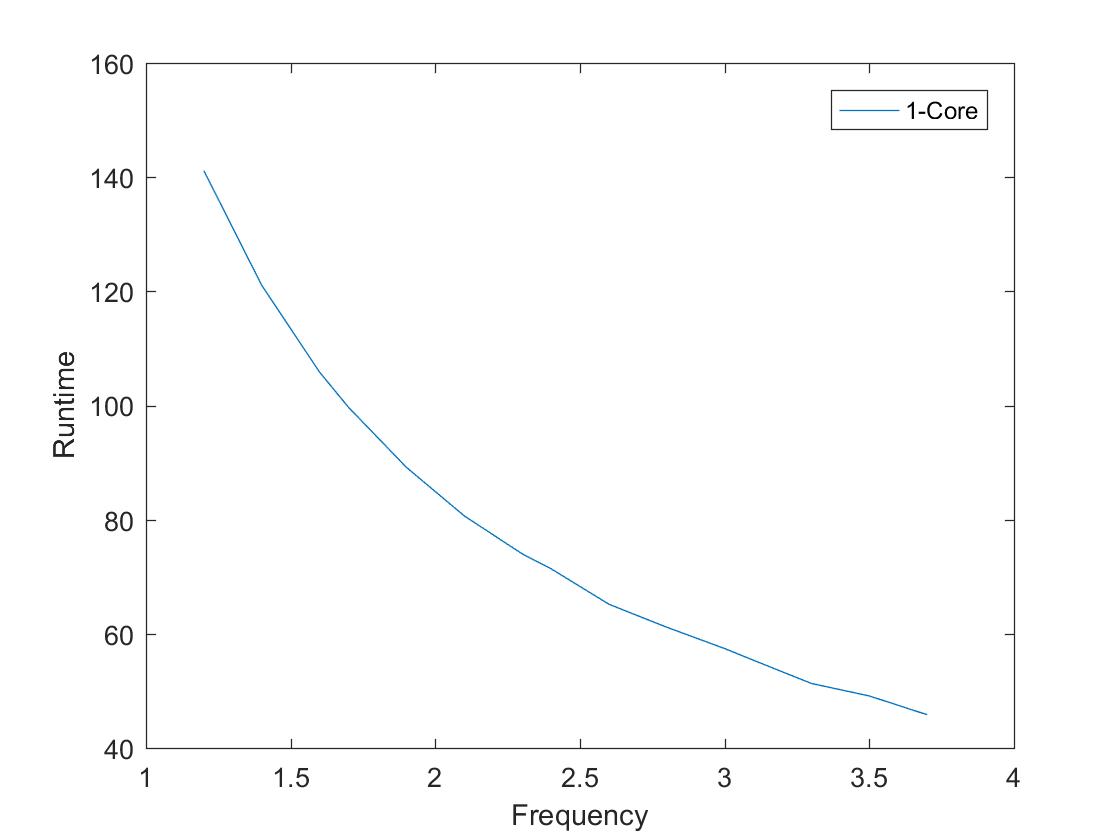
همان طور که انتظار می رود با افزایش فرکانس میزان توان مصرفی در هر سه حالت اندازه گیری شده افزایش می یابد.

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.

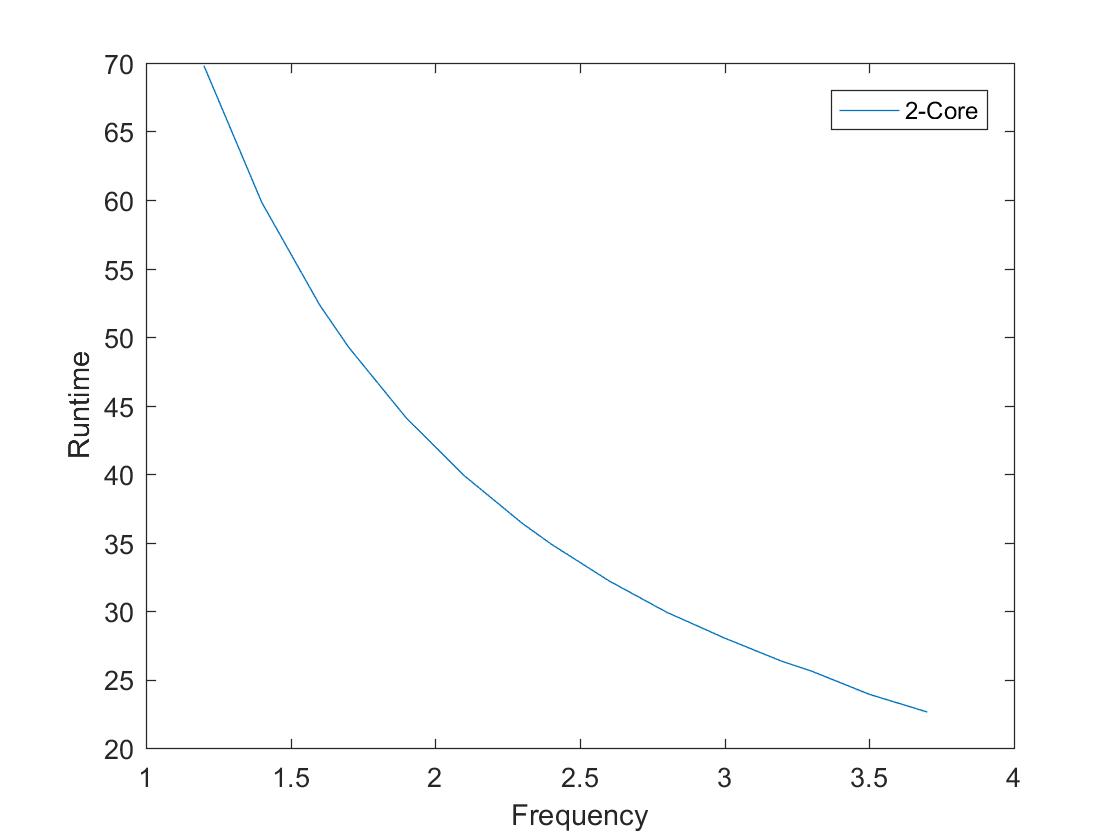
میزان مصرف توان در حالت تک هسته ای با تکنولوژی Hyper-Threading از مصرف در حالت دو هسته ای بیشتر است که علت آن صرف انرژی زیادتر برای موازی سازی در حالت استفاده از Hyper-Threading می باشد که قسمتی از آن به علت استفاده از پردازنده های superscalar است.

ج)

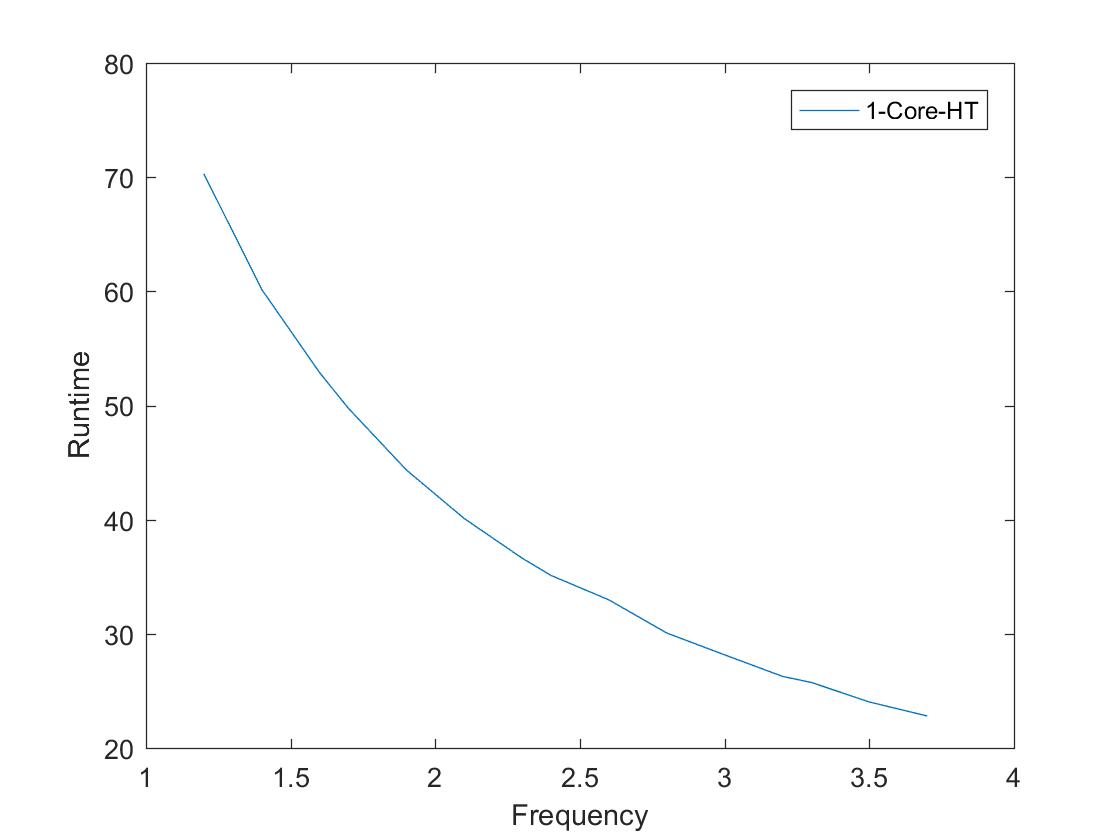
نمودار های زمان اجرا برای 3 حالت تک هسته ای، دو هسته ای و تک هسته ای با تکنولوژی Hyper-Threading در شکل های 5 الی 7 آمده است.



شکل-5 ( زمان اجرا بر حسب فرکانس تک هسته ای)



شکل-6 (زمان اجرا بر حسب فرکانس دو هسته ای)



شکل-7 ( 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 2 حالت کاهش پیدا می کند. با توزیع تسک برروی هسته های مختلف با توجه به موازی انجام شدن محاسبات مختلف اندازه زمان اجرا به صورت قابل ملاحظه ای کاهش پیدا می کند.

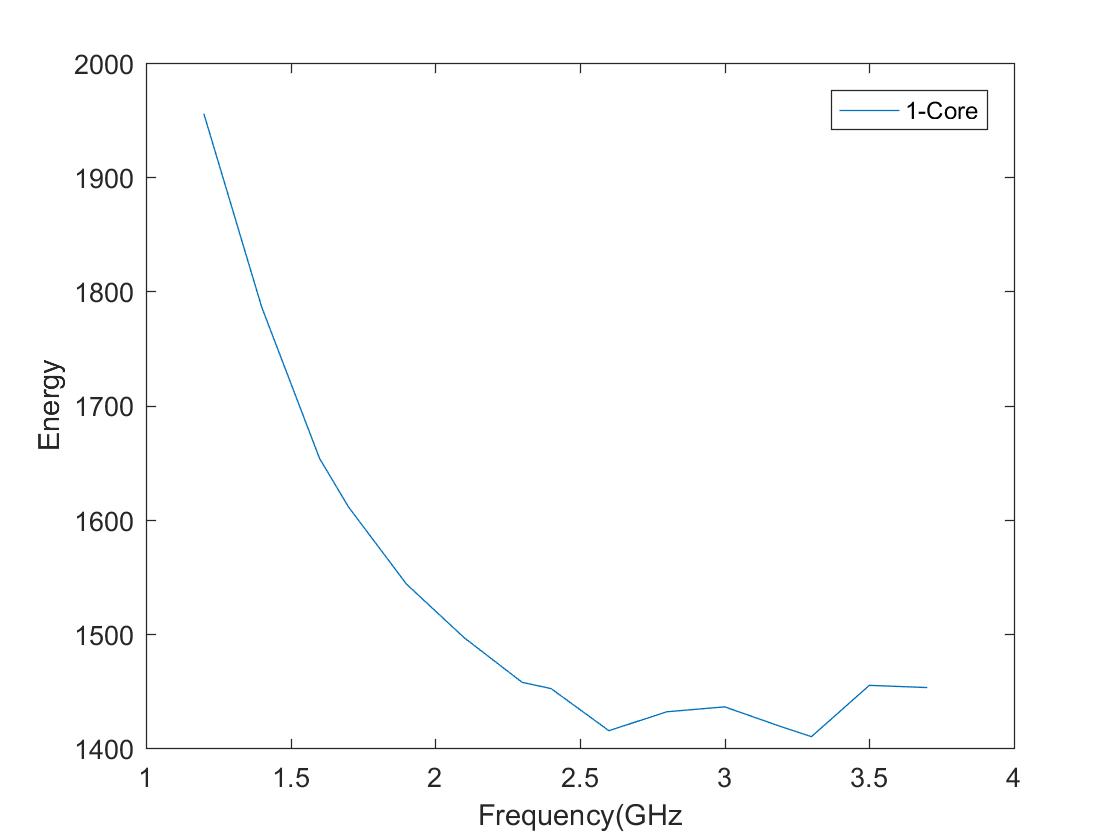
متوسط زمان اجرا برای حالت تک هسته ای 77.7507 ، برای حالت دو هسته ای 38.3398 و برای حالت تک هسته با استفاده از تکنولوژی Hyper-Threading 38.6207 واحد زمانی می باشد. که متوسط درصد کاهش زمان اجرا برای دو حالت دو هسته ای و تک هسته ای با تکنولوژیHT به صورت زیر است.



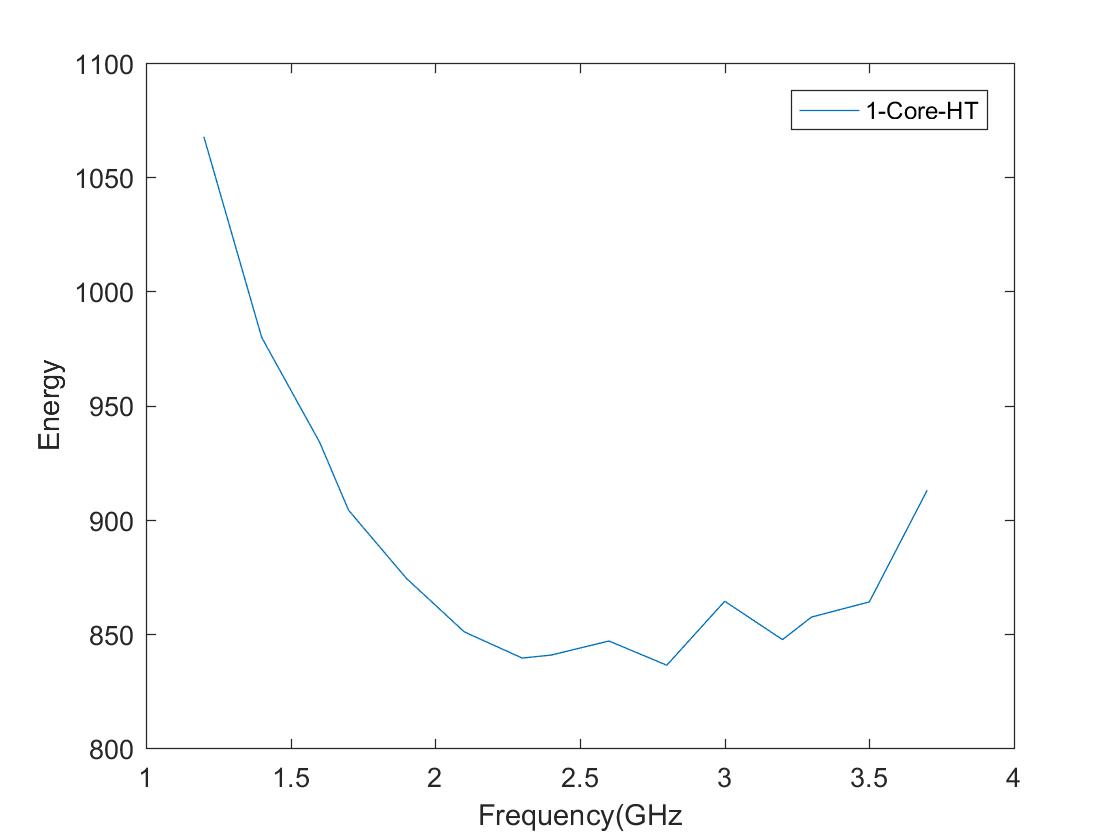
همان طور که مشاهده می شود میزان کاهش زمان اجرا در حالت 2 هسته ای مقدار اندکی از حالت استفاده از تکنولوژی HT بیشتر است که علت آن استفاده از دو پردازنده واقعی به جای دو پردازنده منطقی می باشد.

د)

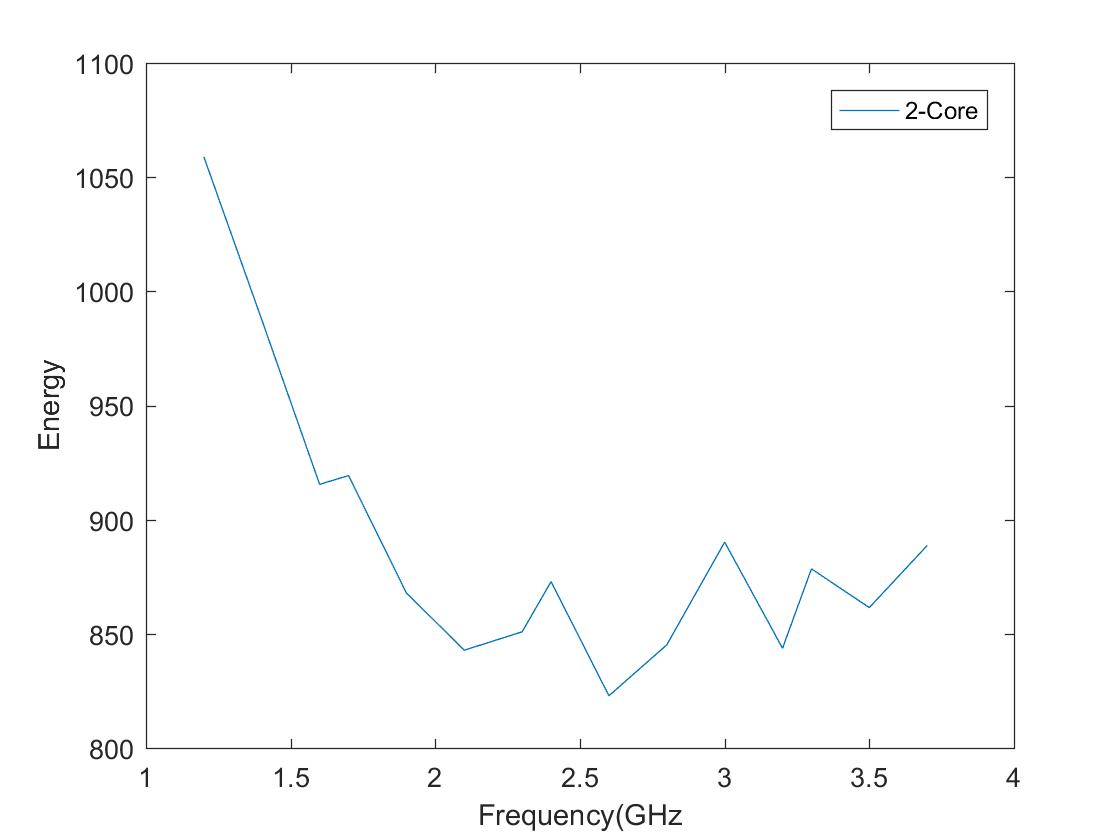
نمودار های میزان مصرف انرژی برای 3 حالت تک هسته ای، دو هسته ای و تک هسته ای با تکنولوژی Hyper-Threading در شکل های 8 الی 10 آمده است.



شکل-8 ( میزان مصرف انرژی بر حسب فرکانس تک هسته ای)



شکل-9 ( میزان مصرف انرژی بر حسب فرکانس تک هسته ای با تکنولوژی HT)

****

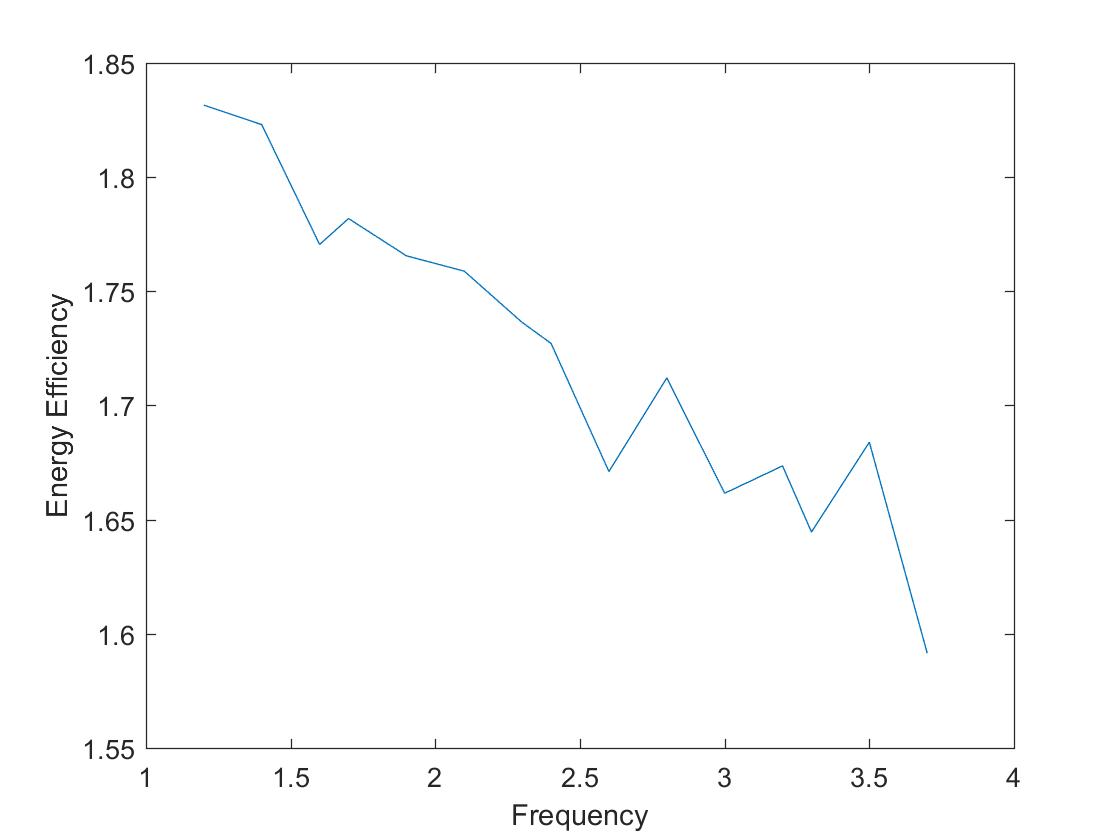
شکل-10 (میزان مصرف انرژی بر حسب فرکانس دو هسته ای)

همان طور که در شکل ها مشاهده می شود میزان مصرف انرژی در ابتدا با افزایش فرکانس به صورت تقریبا 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.

ه)

نمودار بازدهی انرژی در شکل-11 آمده است.



شکل-11 (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 1 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.

2-

الف)

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



میزان مصرف توان منبع تغذیه با توجه به بازده 70 درصدی برای هر سیستم به صورت زیر محاسبه می شود :



ب)

با فرض اینکه 40 درصد از زمان هارد-1 در حالت idle است مقدار متوسط توان مصرفی آن به صورت زیر است :



ج)

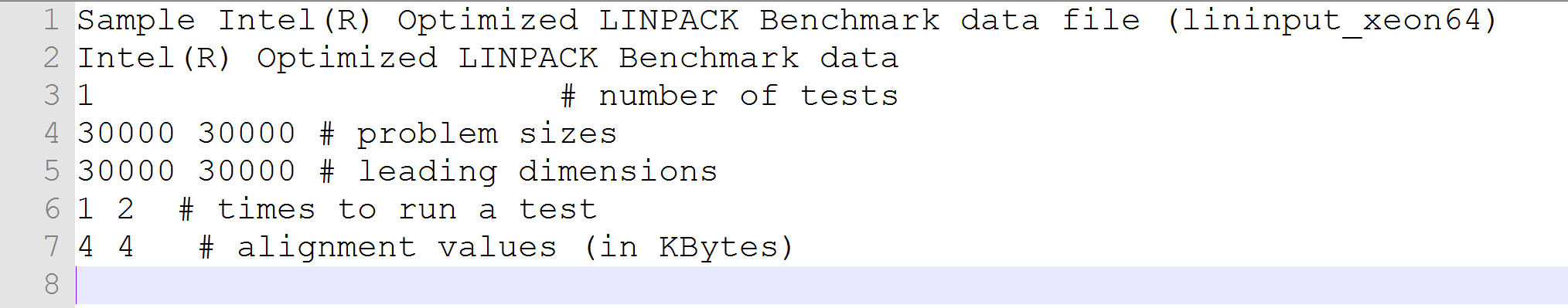
برای اینکه هارد-2 به صورت متوسط 5.6 وات توان مصرف کند باید تساوی زیر برقرار باشد :



با این شرایط اگر این هارد 65 درصد مواقع در حالت idle باشد به اندازه هارد قبلی توان مصرف می کند.

3-

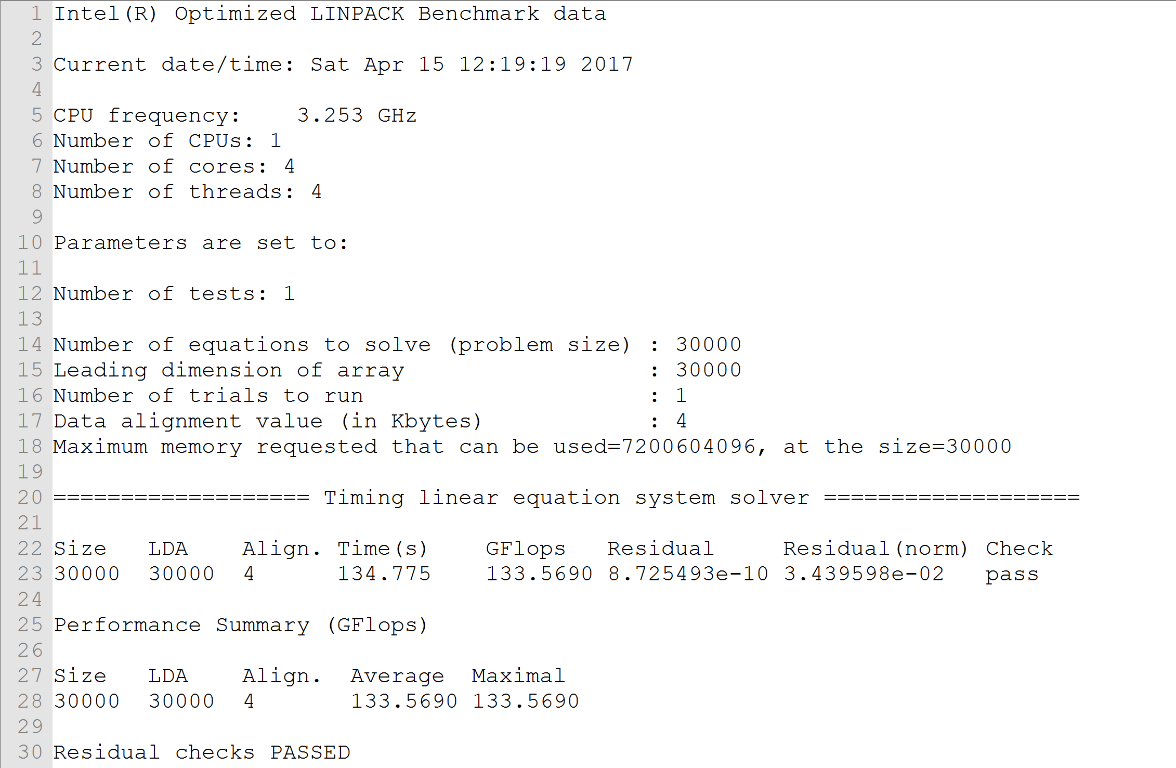
در این قسمت از نرم افزار linpack برای ارزیابی قدرت محاسباتی سیستم استفاده شده است. فایل تنظیمات این برنامه در شکل-12 آمده است.



شکل-12 (تنظیمات برنامه linpack)

ابعاد مساله 30000 در 30000 در نظر گرفته شده است.

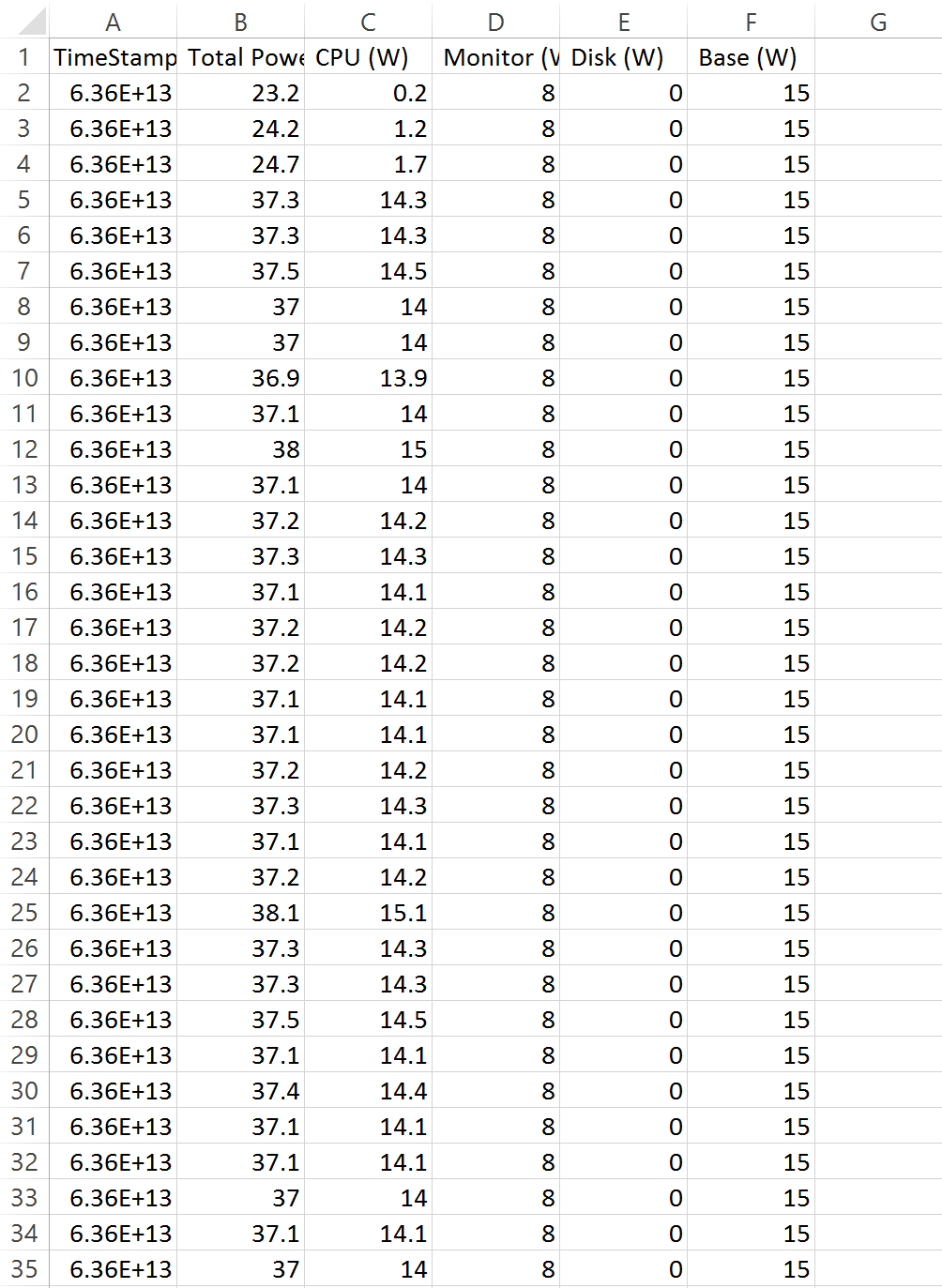
پس از اتمام اجرای برنامه فایل خروجی برنامه تولید و میزان محاسبات را نشان می دهد. این فایل خروجی در شکل-13 آمده است.



شکل-13 (خروجی برنامه linpack)

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

خروجی این برنامه در جدول-1 آمده است.



جدول-1 (خروجی برنامه Joulemeter)

همان طور که در جدول-1 مشاهده می شود 3 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 میمیز شناور به ازای هر وات داریم :



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.

**منابع**

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

**[2]-** Intrl 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%93-rapl

**پیوست (کد برنامه Energy-Calculator)**

**import** os  
**import** sys  
**import** datetime  
DEBUG=**True**warning\_numbers=0  
version=0.1  
logo=**'''  
 \_\_ \_\_ \_  
 / ` / ) //  
 /-- \_\_\_\_ \_ \_\_ \_, \_\_ , / \_\_. // \_.  
(\_\_\_,/ / <\_</\_/ (\_(\_)\_/ (\_/\_ (\_\_/(\_/|\_</\_(\_\_  
 /| /  
 |/ '  
 '''  
def** print\_sign():  
 *'''  
 This function print logo and version* **:return***: None  
 '''* print(logo)  
 print(line(70, **"-"**))  
 print(**"Version :"**+str(version))  
 print(**"By Sepand Haghighi"**)  
 print(line(70,**"-"**))  
**def** line(number,char=**"\*"**):  
 *'''  
 This function return line sign* **:param** *number: number of char* **:type** *number:int* **:param** *char: char item* **:type** *char:str* **:return***: response line as str  
 '''* index=0  
 response=**""  
 while**(index<number):  
 index+=1  
 response+=char  
 **return** response  
**def** convert(num):  
 *'''  
 This function convert number* **:param** *num: input numger* **:type** *num:int* **:return***: output string  
 '''* **for** x **in** [**'wh'**, **'Kwh'**, **'Mwh'**, **'Gwh'**]:  
 **if** num < 1000:  
 **return "%3.1f %s"** % (num, x)  
 num /= 1000

**def** check\_constraint(input\_num,mode=1):  
 *'''  
 This function check for days and hours constraints* **:param** *input\_num: input number* **:type** *input\_num:int* **:param** *mode: mode flag for year and days* **:type** *mode:int* **:return***: input\_number after constraints  
 '''* **if** mode==1:  
 **if** input\_num>24:  
 **return** 24  
 **else**:  
 **if** input\_num>365:  
 **return** 365  
 **return** input\_num  
  
**def** handle\_input(input\_string):  
 *'''  
 This function convert string input to integer and return 0 in error* **:param** *input\_string: input sring* **:type** *input\_string:str* **:return***: integer number  
 '''* **try**:  
 **return** int(input\_string)  
 **except** Exception **as** e:  
 **return** 0  
**def** seperator(input\_string,char=**","**):  
 *'''  
 This function seperate input string to key,wattage,number,hours\_per\_day,days\_per\_year* **:param** *input\_string: input string before seperation* **:type** *input\_string:str* **:return***: list of parameters  
 '''* **try**:  
 seperated\_data = input\_string.split(char)  
 key = seperated\_data[0]  
 wattage = int(seperated\_data[1])  
 number = int(seperated\_data[2])  
 hours\_per\_day = check\_constraint(int(seperated\_data[3]), mode=1)  
 days\_per\_year = check\_constraint(int(seperated\_data[4]), mode=2)  
 **return** [key,wattage,number,hours\_per\_day,days\_per\_year]  
 **except**:  
 **global** warning\_numbers  
 warning\_numbers+=1  
 print(str(warning\_numbers)+**"-Warning : Some issue In this line -->"**,input\_string)  
 print(line(70))  
 **return** [0,0,0,0,0]

**def** find\_ref():  
 *'''  
 This function search for available ref file* **:return***: ref file name  
 '''* **for** item **in** os.listdir():  
 **if** item.find(**".ref"**)!=-1:  
 **return** item  
 **return "NOFILE"**

**def** get\_input():  
 **try**:  
 total\_amount=0  
 file\_name=find\_ref()  
 **if** file\_name==**"NOFILE"**:  
 print(**"There is no ref file"**)  
 sys.exit()  
 file=open(file\_name,**"r"**)  
 output\_file=open(**"energy.out"**,**"w"**)  
 output\_file.write(str(datetime.datetime.now())+**"\n"**)  
 output\_file.write(line(70)+**"\n"**)  
 **for** item **in** file:  
 coefficients=seperator(item)  
 item\_usage=coefficients[1]\*coefficients[2]\*coefficients[3]\*coefficients[4]  
 **if** item\_usage!=0:  
 output\_file.write(coefficients[0]+**" --> "**+convert(item\_usage)+**"\n"**)  
 total\_amount+=item\_usage  
 output\_response=convert(total\_amount)  
 output\_file.write(line(70)+**"\n"**+**"Total :"**+output\_response)  
 file.close()  
 output\_file.close()  
 **return** output\_response  
 **except** Exception **as** e:  
 print(**"Error"**)  
 **if** output\_file.closed==**False**:  
 output\_file.close()  
 **if** DEBUG==**True**:  
 print(str(e))  
  
**if** \_\_name\_\_==**"\_\_main\_\_"**:  
 print\_sign()  
 print(**"Total :"**+get\_input())