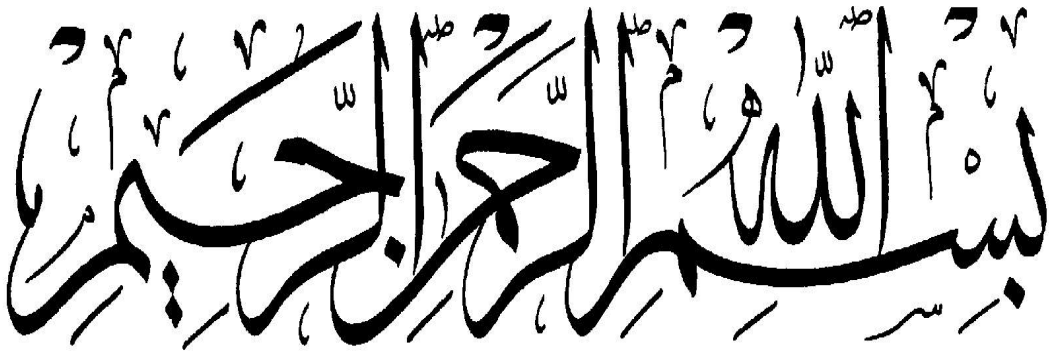
تبدیل مموری کارت‌های SD و MMC به حافظه اجرایی میکروکنترلر (شبه سیستم عامل)

استاد راهنما: [redacted]

تهیه و اجرا: علی تروشه

شماره دانشجویی: [redacted]

رشته کارشناسی ناپیوسته الکترونیک



الشمس و القمر بحسبان

خورشید و ماه با حساب منظمی می‌گردند

الرحمن آیه چهارم

E-mail: ali.tarroosheh@gmail.com

از کسانی که در اجرای این پروژه به من یاری نمودند تشکر می‌کنم

E-mail: ali.tarroosheh@gmail.com

فهرست مطالب

ساختار سخت افزاری و نحوه ارتباط با مموری کارت‌های SD وMMC (صفحه 5)
استاندارد حافظه‌های حجیم یا Microsoft FAT (صفحه 15)
کتابخانه AVR-DOS در کامپایلر BascomAVR (صفحه 33)
Domino مهره‌های به هم پیوسته با وظیفه محدود (صفحه 37)
بخش مدیریت حافظه (صفحه 39)
بخش جرای فایل و در بر گیری کل فضای مموری برای اجرا تا سقف چهار گیگا بایت (صفحه 44)
مبحث پایانی (صفحه 56)
منابع و مأخذ(صفحه 57)

E-mail: ali.tarroosheh@gmail.com

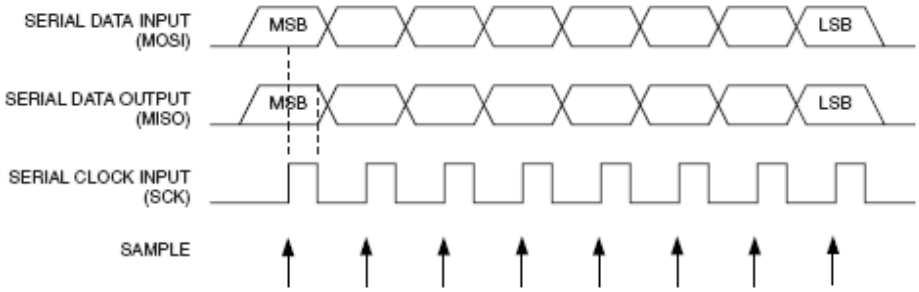
Domino technology

از آنجایی که بسیاری از مطالب به کار برده شده و ذکر شده در این پروژه در طی پنج سال و به مرور زمان ایجاد شده برای اینکه مطلب به درستی قابل فهم باشد آنرا به صورت مجزا در بخش‌های مختلف بررسی می‌کنیم.

قبل از هر چیزی به بررسی مموری کارتهای SD/MMC می‌پردازیم. این نوع حافظه‌ها که از نوع FLASH می‌باشند و به صورت مرسوم در دستگاه‌های قابل حملی مانند MP3 PLAYER ها ، COOL DISK ها و موبایل‌ها استفاده می‌شوند. از خصوصیات بارز آن می‌توان به سرعت بالا و حجم بسیار بالای آن اشاره کرد. این حجم بالا که از چندین مگابایت تا چندین گیگابایت می‌باشد به همان قدر که برای تولید کنندگان این دستگاه‌ها وسوسه انگیز است که برای مهندسان الکترونیک. شاید دلیل عمده این توجه از طرف مهندسان الکترونیک فضای زیادی باشد که در هیچ مدار میکروکنترلری به آن دسترسی ندارند. البته قطعات مختلفی برای رفع این نیاز مانند خانواده AT45DBXXX از شرکت اتمل تولید شد که نوعی FLASH بوده ولی از لحاظ ظرفیت هرگز به گرد SD یا MMC نخواهند رسید. ویژگی بارز دیگر این حافظه قابلیت پشتیبانی از استانداردهای FAT است که با این ویژگی امکان دسترسی به داده‌ها در کامپیوتر بوجود می‌آید.

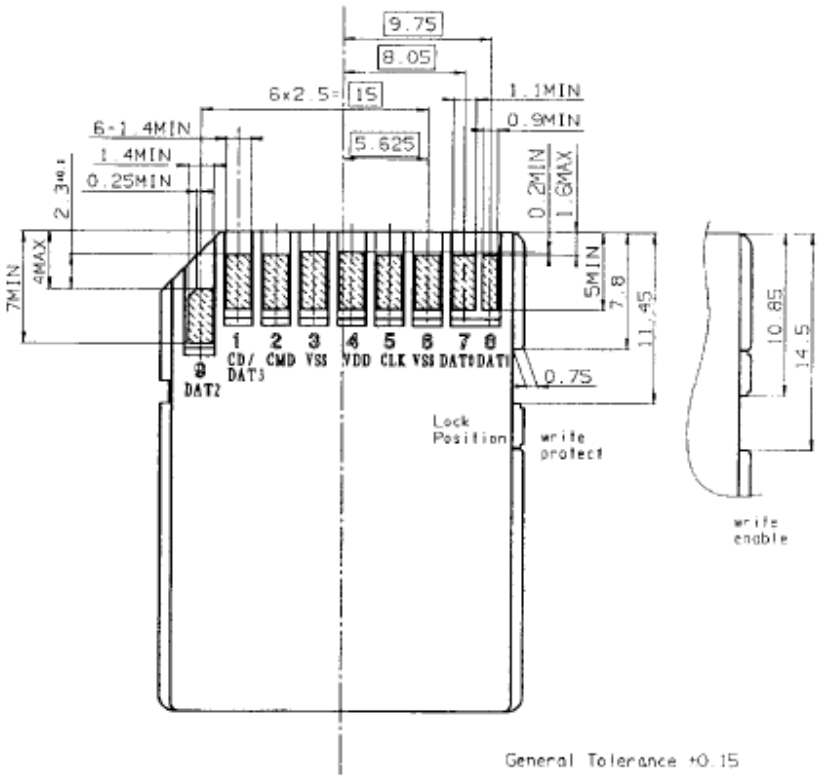
بررسی موری کارت از دید سخت افزاری: SD و MMC تنها از نظر سرعت و حجم بافر با هم تفاوت دارند ولی از نظر نوع فضا و آدرس‌دهی‌ها و... با هم فرقی ندارند. پروتکل ارتباطی مموری به دو شکل IDE MODE یا ارتباط موازی چهاربیتی و نوع SPI MODE یا ارتباط سریال سنکرون می‌باشد که متاسفانه مطالب زیادی درباره نوع اول موجود نیست اما نوع دوم ارتباط مرسومی در میکروکنترلرهاست. این پروتکل ارتباطی به طور معمول دارای سه خط اصلی ارتباطی می‌باشد که یکی از این سه خط وظیفه ارسال داده، دیگری دریافت داده و خط سوم حامل کلاک می‌باشد. در این ارتباط طرفین در دو حالت قرار می‌گیرند یکی MASTER که تولید کننده پالس و حالت دوم SLAVE که هیچگونه اعمالی روی کلاک انجام نمی‌دهد و فقط دریافت کننده آن است. در این ارتباط یک خط نیز حامل سیگنال SS یا SLAVE SELECT می‌باشد که با فعال شدن آن سیستم دریافت کننده این سیگنال به مد SLAVE می‌رود اما این خط به صورت OPTIONAL یا اختیاری است و در اکثر موارد نیازی به آن نیست. در ارتباط با مموری کارت‌ها این خط به نوعی بعنوان CS یا CHIP SELECT به کار می‌رود. در شکل زیر یک نمودار سیگنالی این ارتباط را مشاهده می‌کنید:

E-mail: ali.tarroosheh@gmail.com



نکته مهم دیگر اینکه در این ارتباط می‌توان انتخاب کرد که ابتدا MSB ارسال شود یا LSB که در بحث مموری کارت MSB ابتدا ارسال می‌شود.

MEMORY CARD PIN CONFIGURATION: شکل زیر وضعیت پایه‌های مموری کارت و ابعاد فیزیکی مموری SD را نمایش می‌دهد:



E-mail: ali.tarroosheh@gmail.com

Pin No.	Name	Type[1]	Description
SD Mode			
1	CD/DAT3[2]	I/O[3], PP	Card detect/Data line [Bit 3]
2	CMD	I/O, PP	Command/Response
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DAT0	I/O, PP	Data line [Bit 0]
8	DAT1	I/O, PP	Data line [Bit 1]
9	DAT2	I/O, PP	Data line [Bit 2]
SPI Mode			
1	CS	I	Chip Select (active low)
2	DataIn	I	Host-to-card Commands and Data
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DataOut	O	Card-to-host Data and Status
8	RSV[4]	---	Reserved
9	RSV[5]	---	Reserved

تغذیه مموری بین 2.7 ولت تا 3.6 ولت می‌باشد که برای اطمینان از صحت کار آن توصیه شده با ولتاژ 3.6 ولت تغذیه شود و حافظه بسیار به ولتاژ کاری خود حساس است و امکان آسیب دیدن آن بدلیل اضافه ولتاژ و نویز بسیار زیاد است. فرکانس کاری حافظه در مد SPI تا 50 مگاهرتز بوده و در این پروژه که با میکروکنترلر AVR انجام شده فرکانس کاری حداکثر 8 مگاهرتز می‌باشد.

نکته: برای ارسال هر داده یا دستور به مموری باید ابتدا CS فعال شود بجز مواردی که ذکر خواهد شد.

نکته: این نوع حافظه شامل چندین سکتور است که همه سکتورها در همه انواع مموری کارت 512 بایت است.

E-mail: ali.taroosheh@gmail.com

نحوه ارتباط:

1. **ریست**: برای راه‌اندازی حافظه ابتدا باید از نظر سخت‌افزاری ریست شود. برای اینکار مخالف همه دستورها باید CS غیر فعال شود سپس ده بار عدد 0XFF از طریق SPI به مموری ارسال شود. بلافاصله عدد 0X40 که دستور RESET می‌باشد ارسال و چهار بار عدد 0X00 ارسال شود. در ادامه عدد 0X95 ارسال شود و تا زمانیکه عدد 0X01 از طرف مموری دریافت نشد به مموری عدد 0XFF ارسال شود. در برنامه زیر که به زبان اسمبلی AVR نوشته شده ریست مموری انجام می‌شود:

```
mmc_reset:
sbi mmc,ss
ldi count,10
mmc_reset0:
        ldi temp0,$ff
        rcall send
        dec count
        cpi count,0
brne mmc_reset0
cbi mmc,ss
ldi temp0,$40
rcall send
ldi count,4
mmc_reset1:
        ldi temp0,0
        rcall send
        dec count
        cpi count,0
brne mmc_reset1
```


E-mail: ali.tarroosheh@gmail.com

```
cbi mmc,ss
ldi temp0,$95
rcall send

mmc_reset_ack:
        ldi temp0,$ff
        rcall send
        cpi temp0,$01
brne mmc_reset_ack
sbi mmc,ss
ret
```

که می‌توان تابع SEND را به صورت زیر تعریف کرد:

```
send:
out SPDR,temp0
spi_wait:
sbis spsr,spif
rjmp spi_wait
in temp0,spdr
ret
```

2. **INITIALIZATION MEMORY**: برای INIT کردن مموری ابتدا CS را فعال کرده و عدد 0X41 را ارسال می‌کنیم و بلافاصله چهار بار صفر را ارسال کرده و تا زمانیکه از طرف مموری عدد 0X00 دریافت نشد به مموری عدد 0XFF را ارسال می‌کنیم. برنامه زیر این عمل را انجام می‌دهد:

```
mmc_init:
```


E-mail: ali.tarroosheh@gmail.com

```
cbi mmc,ss
ldi temp0,$41
rcall send
ldi count,4
mmc_init0:
ldi temp0,0
rcall send
dec count
cpi count,0
brne mmc_init0
mmc_init1:
ldi temp0,$ff
rcall send
cpi temp0,0
brne mmc_init1
sbi mmc,ss
ret
```

3. **خواندن یک سکتور از مموری**: بعد از فعال شدن CS و عدد 0X51 را به آن ارسال می-کنیم و به صورت BIG INDIAN دو برابر آدرس مورد نظر را به مموری ارسال کرده و یک بار عدد 0X00 را ارسال کرده و تا زمانیکه عدد 0XFE را دریافت نکردیم به مموری عدد 0XFF را ارسال می‌کنیم. در این لحظه مموری آماده ارسال داده‌های موجود در سکتور مورد نظر بوده و با هر بار ارسال 0XFF یکی از 512 بایت را ارسال می‌کند. در مثال زیر برنامه به گونه‌ای نوشته شده که با هر بار دریافت داده از مموری آنرا روی پورت D قرار می‌دهد:

```
drivereadsector:
cbi mmc,ss
ldi temp0,$51
```


E-mail: ali.tarroosheh@gmail.com

```
rcall send
lds temp0,numofsec+3
rcall send
lds temp0,numofsec+2
rcall send
lds temp0,numofsec+1
rcall send
lds temp0,numofsec
rcall send
ldi temp0,0
rcall send
read0:
ldi temp0,$ff
rcall send
cpi temp0,$fe
brne read0
ldi ZL,0                ;512
ldi ZH,2                ;512
recive1:
ldi temp0,$ff
cbi hand,num
rcall send
out PORTD,temp0
sbi hand,num
sbiw ZH:ZL,1
cpi ZL,0
```

```
brne recive1
cpi ZH,0
brne recive1
ldi temp0,$ff
rcall send
ldi temp0,$ff
rcall send
cbi hand,num
sbi mmc,ss
ret
```

4. **نوشتن در یک سکتور**: برای نوشتن در یک سکتور ابتدا عدد 0X58 ارسال شده سپس دو برابر آدرس سکتور مورد نظر ارسال و سه بار 0XFF و یک بار 0XFE ارسال شده در این لحظه می‌توان 512 بایت داده را ارسال کرد بعد از این داده‌ها دو بار 0XFF ارسال می‌شود و می‌توان CS را غیر فعال کرد.

مثال زیر یک برنامه ساده تحت KEIL C برای میکروکنترلر AMR سری فیلیپس به شماره LPC2106 می‌باشد که تمام اعمال فوق را که تا کنون بحث کردیم انجام می‌دهد:

```
#include "LPC21xx.h"

#define SPIF (1<<7)
#define spi_wait (!(S0SPSR & SPIF))
const unsigned char mmc_init[]={0x41,0x00,0x00,0x00,0x00,0x95};
const unsigned char mmc_rst[]={0x40,0x00,0x00,0x00,0x00,0x95};

//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
void init(void){
        unsigned char acc;
        char cnt;
        IOSET0=128;
                for(cnt=0;cnt<10;cnt++){
                        S0SPDR=0xFF;
                        while(spi_wait);
                }
        IOCLR0=128;
        for (cnt=0;cnt<6;cnt++){
        S0SPDR=mmc_init[cnt];
        while(spi_wait);}
        for(;;){
```

```
                S0SPDR=0xFF;
                while(spi_wait);
                if (S0SPDR==0){IOSET0=128;break;}
        }
}
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
void rst(void){
        unsigned char acc;
        char cnt;
        IOSET0=128;
                for(cnt=0;cnt<10;cnt++){
                        S0SPDR=0xFF;
                        while(spi_wait);
                }
        IOCLR0=128;
        for (cnt=0;cnt<6;cnt++){
        S0SPDR=mmc_rst[cnt];
        while(spi_wait);}
        for(;;){
                S0SPDR=0xFF;
                while(spi_wait);
                if (S0SPDR==1){IOSET0=128;break;}
        }
}
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
void reader(unsigned long position,char *sector){
unsigned char i[4]={0};
int cnt;
position=position*2;

for(cnt=3;cnt>0;cnt--){
i[cnt-1]=position%256;
position=position/256;
}


IOCLR0=128;
S0SPDR=0x51;
while(spi_wait);
for (cnt=0;cnt<4;cnt++){
S0SPDR=i[cnt];
while(spi_wait);
}

for(;;){
S0SPDR=0xFF;
while(spi_wait);
if (S0SPDR==0xFE){break;}}

for(cnt=0;cnt<512;cnt++){
S0SPDR=0xFF;
while(spi_wait);
sector[cnt]=S0SPDR;
```

```
}
S0SPDR=0xFF;
while(spi_wait);
S0SPDR=0xFF;
while(spi_wait);
IOSET0=128;

}
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
void writer(unsigned long position,char *sector){
unsigned char i[4]={0};
int cnt;
position=position*2;

for(cnt=3;cnt>0;cnt--){
i[cnt-1]=position%256;
position=position/256;
}


IOCLR0=128;
S0SPDR=0x58;
while(spi_wait);
for (cnt=0;cnt<4;cnt++){
S0SPDR=i[cnt];
while(spi_wait);
}

S0SPDR=0xFF;while(spi_wait);
S0SPDR=0xFF;while(spi_wait);
S0SPDR=0xFF;while(spi_wait);
S0SPDR=0xFE;while(spi_wait);


for(cnt=0;cnt<512;cnt++){
S0SPDR=sector[cnt];
while(spi_wait);
}
S0SPDR=0xFF;while(spi_wait);
S0SPDR=0xFF;while(spi_wait);
IOSET0=128;

}
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

MICROSOFT FAT

یکی از خارق العاده ترین تکنولوژی‌هایی که شرکت مایکروسافت ارائه داده FAT می‌باشد این استاندارد که برای مکان حافظه‌های بسیار حجیم بوجود آمد در ابتدا با استاندارد FAT12 ارائه شد که قابلیت آدرس دهی 12 بیتی کلاسترها را داشت و تا 16 مگابایت مموری یا هارددیسک را پشتیبانی می‌کرد ولی امروزه بدلیل حجم بالای ظرفیت این نوع استاندارد منسوخ شده است. در ادامه نمونه قویتری با نام FAT16 را ارائه کرد که قابلیت آدرس دهی 16 بیتی کلاسترها را داشت و تا ظرفیت 4 گیگا بایت را پشتیبانی می‌کرد این نمونه نیز با روی کار آمدن فضاهای بسیار بالا در هارد دیسکها کنار رفته و فقط در سیستم‌های قابل حملی مانند MP3 PLAYER و COOL DISK ها استفاده می-شود. این نمونه بدلیل پائین بودن تعداد سکتور به‌ازای هر کلاستر در سیستم‌های قدیمی سرعت کمتری را داراست زیرا در حجم برابر به محاسبات بیشتری نسبت به نمونه جدیدتر خود نیاز دارد تا ادامه فایل ذخیره شده را در حافظه پیدا کند. به این دلیل و محدودیت ظرفیت نمونه دیگری ارائه شد که با نام FAT32 قابلیت آدرس دهی 32 بیتی کلاسترها را دارا بود. این نمونه که تا چندی پیش در کامپیوترهای شخصی حرف اول را در مورد مدیریت هارددیسک‌ها می‌زد تا ظرفیت 32 گیگابایت حافظه را پشتیبانی می‌کرد و از سرعت بالایی نسبت به همه نمونه‌های قبلی برخوردار بود. اما با نمونه آخر با نام NTFS فعلا تمام حرف‌ها و حدیث‌ها تمام شد، این استاندارد که با نمونه‌های قبل فرق داشته و ظرفیت 2 ترابایت را در حافظه پشتیبانی می‌کند و بسیار جمع و جورتر از همه طراحی شد و از ریخت و پاش فضای حافظه بسیار جلوگیری شده و از لحاظ سرعت و میزان محاسبات مورد نیاز بسیار بهینه-سازی شد.

در ادامه به بحث در مورد بعضی از مفاهیم موجود در این استاندارد و کلمات کلیدی آن می‌پردازیم و با توضیح لازم تفاوت‌های موجود بین نمونه‌های مختلف را شرح می‌دهیم. البته چون مبحث اصلی مموری کارت‌های SD و MMC می‌باشد و بغیر از ویندوز VISTA بقیه ویندوزها قابلیت پشتیبانی NTFS را روی این نوع حافظه ندارند اینجانب به شخصه از کار روی این استاندارد صرف نظر کرده و بعلاوه در مورد FAT12 بدلیل اینکه در بازار مموری با ظرفیت 16 مگابایت موجود نیست تحقیقی بعمل نیامد.

مفاهیم کلیدی:

1. SECTOR: فضای مموری کارت‌ها یا هارد دیسک‌ها به تعدادی سکتور تقسیم می‌شوند که برای مموری کارت‌های SD و MMC سکتور شامل 512 بایت می‌باشد و مموری کارت در هنگام نوشتن و خواندن بطور مستقیم با بایت‌ها سر و کار ندارد بلکه با سکتورها کار می‌کند و

E-mail: ali.taroosheh@gmail.com

به این ترتیب برای اینکه بتوان بشکل معمول با مموری کار کرد بهتر است که میکروکنترلری استفاده کرد که از 512 بایت بیشتر SRAM داشته باشد.

2. CLUSTER: از دید سخت افزاری کلاستر وجود خارجی ندارد و فقط در استاندارد FAT فایل سیستم بجای اینکه با سکتورها کار کند که تعداد زیادی می‌باشند با کلاستر کار می‌کند، و کلاستر به مجموعه‌ای از سکتورها اطلاق می‌شود که برای هر نوع FAT اعم از FAT12 ، FAT16 ، FAT32 و NTFS متفاوت و وابسته به ظرفیت حافظه می‌باشد. در جدول زیر این مطلب به روشنی مشخص شده:

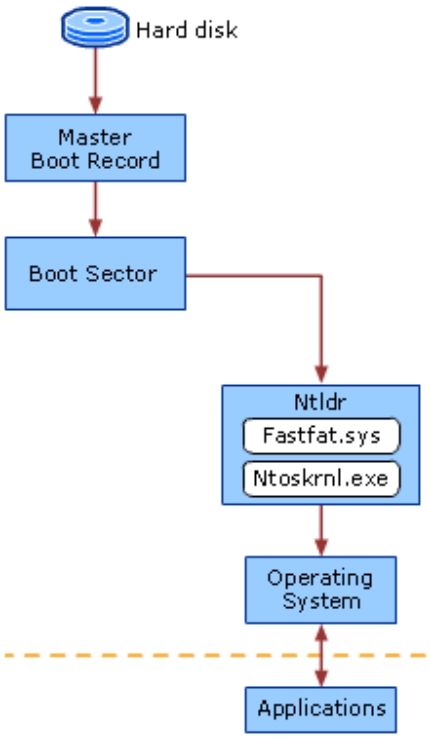
Volume size	FAT16 cluster size	FAT32 cluster size	NTFS cluster size
7 MB–16 MB	2 KB	Not supported	512 bytes
17 MB–32 MB	512 bytes	Not supported	512 bytes
33 MB–64 MB	1 KB	512 bytes	512 bytes
65 MB–128 MB	2 KB	1 KB	512 bytes
129 MB–256 MB	4 KB	2 KB	512 bytes
257 MB–512 MB	8 KB	4 KB	512 bytes
513 MB–1,024 MB	16 KB	4 KB	1 KB
1,025 MB–2 GB	32 KB	4 KB	2 KB
2 GB–4 GB	64 KB	4 KB	4 KB
4 GB–8 GB	Not supported	4 KB	4 KB
8 GB–16 GB	Not supported	8 KB	4 KB
16 GB–32 GB	Not supported	16 KB	4 KB
32 GB–2 TB	Not supported	Not supported	4 KB

از آنجایی که هر سکتور در این نوع مموری 512 بایت است پس به طور مثال برای ظرفیت 129 مگابایت تا 256 مگابایت در FAT32 که کلاستر سایز 2 کیلوبایت ذکر شده یعنی 4 سکتور.

3. MBR (MASTER BOOT RECORD): MBR یک کد اجرایی کوچکی است که در بخشی از حافظه قرار می‌گیرد تا سیستم از آن به اصطلاح BOOT شود. در شکل زیر نحوه بوت شدن یک سیستم عامل را ملاحظه می‌کنید:

E-mail: ali.taroosheh@gmail.com

FAT Architecture



4. اجزای FAT: در جدول زیر این اجزا مشخص است:

FAT Volume Components

Component	Description
Boot Sector	Contains the BIOS parameter block that stores information about the layout of the volume and the file system structures, as well as the boot code that loads Windows Server 2003.
Reserved Sectors	The number of sectors that precede the start of the first FAT, including the boot sector.
FAT 1	Original FAT.
FAT 2 (Duplicate)	Backup copy of the FAT.
Root folder	Describes the files and folders in the root of the partition.
Other folders and all files	Contains the data for the files and folders within the file system.

E-mail: ali.tarroosheh@gmail.com

معمولا سکتور صفر در مموری کارت‌ها شامل BOOT SECTOR بوده و بعد از آن یک فضایی رزرو شده و فضای FAT اول قرار دارد و FAT دوم برای ریکاوری و BACKUP دقیقا شبیه به FAT اول قرار می‌گیرد.

5. ROOT DIRECTORY: به شاخه اصلی مموری کارت ROOT DIRECTORY می‌گویند که در آن آدرس فایل‌ها و فولدرهای موجود و نام و مشخصاتشان نوشته شده است.

*. در ادامه برای روشن شدن موضوع هر FAT را به صورت جداگانه بررسی می‌کنیم.

FAT32: همانطور که گفته شد در مموری کارت‌ها سکتور بوت معمولا در سکتور صفر قرار دارد. این سکتور شامل اطلاعات اصلی فضای حافظه است. در شکل زیر اطلاعات موجود در این سکتور را که برای حافظه خاصی نوشته شده مشاهده می‌کنید:

```
Physical Sector:  Cyl  0,  Side  1,  Sector  1
00000000:  EB 3C 90 4D 53 44 4F 53  -  35 2E 30 00 02 40 01 00    .<.MSDOS5.0. .@..
00000010:  02 00 02 00 00 F8 FC 00  -  3F 00 40 00 3F 00 00 00    ........?.@.?...
00000020:  01 F0 3E 00 80 00 29 A8  -  8B 36 52 4E 4F 20 4E 41    ..>...)..6RNO NA
00000030:  4D 45 20 20 20 20 46 41  -  54 31 36 20 20 20 33 C0    ME    FAT16   3.
00000040:  8E D0 BC 00 7C 68 C0 07  -  1F A0 10 00 F7 26 16 00    ....|h.....&..
00000050:  03 06 0E 00 50 91 B8 20  -  00 F7 26 11 00 8B 1E 0B    ....P.. ..&.....
00000060:  00 03 C3 48 F7 F3 03 C8  -  89 0E 08 02 68 00 10 07    ...H........h...
00000070:  33 DB 8F 06 13 02 89 1E  -  15 02 0E E8 90 00 72 57    3.............rW
00000080:  33 DB 8B 0E 11 00 8B FB  -  51 B9 0B 00 BE DC 01 F3    3.......Q.......
00000090:  A6 59 74 05 83 C3 20 E2  -  ED E3 37 26 8B 57 1A 52    .Yt... ...7&.W.R
000000A0:  B8 01 00 68 00 20 07 33  -  DB 0E E8 48 00 72 28 5B    ...h. .3...H.r([
000000B0:  8D 36 0B 00 8D 3E 0B 02  -  1E 8F 45 02 C7 05 F5 00    .6...>....E.....
000000C0:  1E 8F 45 06 C7 45 04 0E  -  01 8A 16 24 00 EA 03 00    ..E..E.....$....
000000D0:  00 20 BE 86 01 EB 03 BE  -  A2 01 E8 09 00 BE C1 01    . ..............
000000E0:  E8 03 00 FB EB FE AC 0A  -  C0 74 09 B4 0E BB 07 00    .........t......
000000F0:  CD 10 EB F2 C3 50 4A 4A  -  A0 0D 00 32 E4 F7 E2 03    .....PJJ...2....
00000100:  06 08 02 83 D2 00 A3 13  -  02 89 16 15 02 58 A2 07    .............X..
00000110:  02 A1 13 02 8B 16 15 02  -  03 06 1C 00 13 16 1E 00    ................
00000120:  F7 36 18 00 FE C2 88 16  -  06 02 33 D2 F7 36 1A 00    .6........3..6..
00000130:  88 16 25 00 A3 04 02 A1  -  18 00 2A 06 06 02 40 3A    ..%.......*...@:
00000140:  06 07 02 76 05 A0 07 02  -  32 E4 50 B4 02 8B 0E 04    ...v....2.P.....
00000150:  02 C0 E5 06 0A 2E 06 02  -  86 E9 8B 16 24 00 CD 13    ............$...
00000160:  0F 83 05 00 83 C4 02 F9  -  CB 58 28 06 07 02 76 11    .........X(...v.
00000170:  01 06 13 02 83 16 15 02  -  00 F7 26 0B 00 03 D8 EB    ..........&.....
00000180:  90 A2 07 02 F8 CB 42 4F  -  4F 54 3A 20 43 6F 75 6C    ......BOOT: Coul
00000190:  64 6E 27 74 20 66 69 6E  -  64 20 4E 54 4C 44 52 0D    dn't find NTLDR.
000001A0:  0A 00 42 4F 4F 54 3A 20  -  49 2F 4F 20 65 72 72 6F    ..BOOT: I/O erro
000001B0:  72 20 72 65 61 64 69 6E  -  67 20 64 69 73 6B 0D 0A    r reading disk..
000001C0:  00 50 6C 65 61 73 65 20  -  69 6E 73 65 72 74 20 61    .Please insert a
000001D0:  6E 6F 74 68 65 72 20 64  -  69 73 6B 00 4E 54 4C 44    nother disk.NTLD
000001E0:  52 20 20 20 20 20 20 00  -  00 00 00 00 00 00 00 00    R       ........
000001F0:  00 00 00 00 00 00 00 00  -  00 00 00 00 00 00 55 AA    ..............U.
```

توضیحات: از بایت صفر تا بایت دهم این فضا اطلاعاتی از شرکت مورد نظر وجود دارد که در اینجا MSDOS5.0 را ملاحظه می‌کنید.

برای درک بیشتر با سیستم طراحی شده سکتور صفر را بررسی می‌کنیم که از یک مموری 128 مگابایتی با fat32 خوانده شده. در داده‌های زیر هر خط یکی از 512 بایت موجود است:

point1:(ë);235	point3:( );144	point5:(S);83	point7:(O);79
point2:(X);88	point4:(M);77	point6:(D);68	point8:(S);83

E-mail: ali.taroosheh@gmail.com

point9:(5);53
point10:(.);46
point11:(0);48
point12:();0
point13:(_);2
point14:(_);2
point15:($);36
point16:();0
point17:(_);2
point18:();0
point19:();0
point20:();0
point21:();0
point22:(ø);248
point23:();0
point24:();0
point25:(?);63
point26:();0
point27:(ÿ);255
point28:();0
point29:();0
point30:();0
point31:();0
point32:();0
point33:();0
point34:(Å);197
point35:(_);3
point36:();0
point37:(¾);190
point38:(_);3
point39:();0
point40:();0
point41:();0
point42:();0
point43:();0
point44:();0
point45:(_);2
point46:();0
point47:();0
point48:();0
point49:(_);1
point50:();0
point51:(_);6
point52:();0
point53:();0
point54:();0
point55:();0
point56:();0
point57:();0
point58:();0
point59:();0
point60:();0
point61:();0
point62:();0
point63:();0
point64:();0
point65:();0
point66:();0
point67:());41
point68:(_);6
point69:(Œ);140
point70:(\);92
point71:(_);28
point72:(N);78
point73:(O);79
point74:( );32
point75:(N);78
point76:(A);65
point77:(M);77
point78:(E);69
point79:( );32
point80:( );32
point81:( );32
point82:( );32
point83:(F);70
point84:(A);65
point85:(T);84
point86:(3);51
point87:(2);50
point88:( );32
point89:( );32
point90:( );32
point91:(3);51
point92:(É);201
point93:(Ž);142
point94:(Ñ);209
point95:(¼);188
point96:(ô);244
point97:({);123
point98:(Ž);142
point99:(Á);193
point100:(Ž);142
point101:(Ù);217
point102:(½);189
point103:();0
point104:(|);124
point105:(¯);136
point106:(N);78
point107:(_);2
point108:(Š);138
point109:(V);86
point110:(@);64
point111:(´);180
point112:(_);8
point113:(Í);205
point114:(_);19
point115:(s);115
point116:(_);5
point117:(¹);185
point118:(ÿ);255
point119:(ÿ);255
point120:(Š);138
point121:(ñ);241
point122:(f);102
point123:(_);15
point124:();182
point125:(Æ);198
point126:(@);64
point127:(f);102
point128:(_);15
point129:();182
point130:(Ñ);209
point131:(€);128
point132:(â);226
point133:(?);63
point134:(÷);247
point135:(â);226
point136:(†);134
point137:(Í);205
point138:(À);192
point139:(í);237
point140:(_);6
point141:(A);65
point142:(f);102
point143:(_);15
point144:(•);183
point145:(É);201
point146:(f);102
point147:(÷);247
point148:(á);225
point149:(f);102
point150:(‰);137
point151:(F);70
point152:(ø);248
point153:(ƒ);131
point154:(~);126
point155:(_);22
point156:();0
point157:(u);117
point158:(8);56
point159:(ƒ);131
point160:(~);126
point161:(*);42
point162:();0
point163:(w);119
point164:(2);50
point165:(f);102
point166:(‹);139
point167:(F);70
point168:(_);28
point169:(f);102
point170:(ƒ);131
point171:(À);192
point172:();12
point173:(»);187
point174:();0
point175:(€);128
point176:(¹);185
point177:(_);1
point178:();0
point179:(è);232
point180:(+);43
point181:();0
point182:(é);233
point183:(H);72
point184:(_);3
point185:( );160
point186:(ú);250
point187:(});125
point188:(´);180
point189:(});125
point190:(‹);139
point191:(ð);240
point192:();172
point193:(„);132
point194:(À);192
point195:(t);116
point196:(_);23
point197:(<);60
point198:(ÿ);255
point199:(t);116
point200:( );9
point201:(´);180
point202:(_);14
point203:(»);187
point204:(_);7
point205:();0
point206:(Í);205
point207:(_);16
point208:(ë);235
point209:(î);238
point210:( );160
point211:(û);251
point212:(});125
point213:(ë);235
point214:(å);229
point215:( );160
point216:(ù);249
point217:(});125
point218:(ë);235
point219:(à);224
point220:(¯);152
point221:(Í);205
point222:(_);22
point223:(Í);205
point224:(_);25
point225:(f);102
point226:(`);96
point227:(f);102
point228:(;);59
point229:(F);70
point230:(ø);248
point231:(_);15
point232:(,);130
point233:(J);74
point234:();0
point235:(f);102
point236:(j);106
point237:();0
point238:(f);102
point239:(P);80
point240:(_);6
point241:(S);83
point242:(f);102
point243:(h);104
point244:(_);16
point245:();0
point246:(_);1
point247:();0
point248:(€);128
point249:(~);126
point250:(_);2
point251:();0
point252:(_);15
point253:(…);133
point254:( );32
point255:();0
point256:(´);180

E-mail: ali.tarroosheh@gmail.com

point257:(A);65
point258:(»);187
point259:(ª);170
point260:(U);85
point261:(Š);138
point262:(V);86
point263:(@);64
point264:(Í);205
point265:(_);19
point266:(_);15
point267:(‚);130
point268:(_);28
point269:();0
point270:( );129
point271:(û);251
point272:(U);85
point273:(ª);170
point274:(_);15
point275:(…);133
point276:(_);20
point277:();0
point278:(ö);246
point279:(Á);193
point280:(_);1
point281:(_);15
point282:(„);132
point283:(
);13
point284:();0
point285:(þ);254
point286:(F);70
point287:(_);2
point288:(´);180
point289:(B);66
point290:(Š);138
point291:(V);86
point292:(@);64
point293:(‹);139
point294:(ô);244
point295:(Í);205
point296:(_);19
point297:(°);176
point298:(ù);249
point299:(f);102
point300:(X);88
point301:(f);102
point302:(X);88
point303:(f);102
point304:(X);88
point305:(f);102
point306:(X);88
point307:(ë);235
point308:(*);42
point309:(f);102
point310:(3);51
point311:(Ò);210
point312:(f);102
point313:(_);15
point314:(•);183
point315:(N);78
point316:(_);24
point317:(f);102
point318:(÷);247
point319:(ñ);241
point320:(þ);254
point321:(Â);194
point322:(Š);138
point323:(Ê);202
point324:(f);102
point325:(‹);139
point326:(Đ);208
point327:(f);102
point328:(Á);193
point329:(ê);234
point330:(_);16
point331:(÷);247
point332:(v);118
point333:(_);26
point334:(†);134
point335:(Ö);214
point336:(Š);138
point337:(V);86
point338:(@);64
point339:(Š);138
point340:(è);232
point341:(À);192
point342:(ä);228
point343:(_);6
point344:();10
point345:(Ì);204
point346:(‚);184
point347:(_);1
point348:(_);2
point349:(Í);205
point350:(_);19
point351:(f);102
point352:(a);97
point353:(_);15
point354:(‚);130
point355:(T);84
point356:(ÿ);255
point357:( );129
point358:(Ã);195
point359:();0
point360:(_);2
point361:(f);102
point362:(@);64
point363:(I);73
point364:(_);15
point365:(…);133
point366:(q);113
point367:(ÿ);255
point368:(Ã);195
point369:(N);78
point370:(T);84
point371:(L);76
point372:(D);68
point373:(R);82
point374:( );32
point375:( );32
point376:( );32
point377:( );32
point378:( );32
point379:( );32
point380:();0
point381:();0
point382:();0
point383:();0
point384:();0
point385:();0
point386:();0
point387:();0
point388:();0
point389:();0
point390:();0
point391:();0
point392:();0
point393:();0
point394:();0
point395:();0
point396:();0
point397:();0
point398:();0
point399:();0
point400:();0
point401:();0
point402:();0
point403:();0
point404:();0
point405:();0
point406:();0
point407:();0
point408:();0
point409:();0
point410:();0
point411:();0
point412:();0
point413:();0
point414:();0
point415:();0
point416:();0
point417:();0
point418:();0
point419:();0
point420:();0
point421:();0
point422:();0
point423:();0
point424:();0
point425:();0
point426:();0
point427:();0
point428:();0
point429:(
);13
point430:();10
point431:(R);82
point432:(e);101
point433:(m);109
point434:(o);111
point435:(v);118
point436:(e);101
point437:( );32
point438:(d);100
point439:(i);105
point440:(s);115
point441:(k);107
point442:(s);115
point443:( );32
point444:(o);111
point445:(r);114
point446:( );32
point447:(o);111
point448:(t);116
point449:(h);104
point450:(e);101
point451:(r);114
point452:( );32
point453:(m);109
point454:(e);101
point455:(d);100
point456:(i);105
point457:(a);97
point458:(.);46
point459:(ÿ);255
point460:(
);13
point461:();10
point462:(D);68
point463:(i);105
point464:(s);115
point465:(k);107
point466:( );32
point467:(e);101
point468:(r);114
point469:(r);114
point470:(o);111
point471:(r);114
point472:(ÿ);255
point473:(
);13
point474:();10
point475:(P);80
point476:(r);114
point477:(e);101
point478:(s);115
point479:(s);115
point480:( );32
point481:(a);97
point482:(n);110
point483:(y);121
point484:( );32
point485:(k);107
point486:(e);101
point487:(y);121
point488:( );32
point489:(t);116
point490:(o);111
point491:( );32
point492:(r);114
point493:(e);101
point494:(s);115
point495:(t);116
point496:(a);97
point497:(r);114
point498:(t);116
point499:(
);13

E-mail: ali.tarroosheh@gmail.com

point500:();10 point501:();0 point502:();0 point503:();0	point504:();0 point505:();0 point506:();172 point507:(Ë);203	point508:(Ø);216 point509:();0 point510:();0 point511:(U);85	point512:(ª);170

همانطور که گفته شد ده بایت اول این سکتور اطلاعاتی از شرکت پیاده کننده وجود دارد که در جدول بالا چون داده‌ها از یک شروع شده به این دلیل آدرس مربوط به این اطلاعات از بایت اول تا بایت یازدهم است. اما بایت‌های دیگر:

Byte per sector: بایت دوازدهم و سیزدهم تعداد هر بایت در هر سکتور را مشخص می‌کند که در اینجا با توجه به توضیحات داده شده همیشه عدد 512 قرار دارد.

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x0B	2 bytes	00 02	**Bytes Per Sector.** The size of a hardware sector. Valid decimal values for this field are 512, 1024, 2048, and 4096.

Sector per cluster: این فضای یک بایتی تعداد سکتور به‌ازای هر کلاستر را معرفی می‌کند.

0x0D	1 byte	10	**Sectors Per Cluster.**The number of sectors in a cluster. The default cluster size for a volume depends on the volume size. Valid decimal values for this field are 1, 2, 4, 8, 16, 32, 64, and 128. The Windows Server 2003 implementation of FAT32 allows for creation of volumes up to a maximum of 32 GB. However, larger volumes created by other operating systems (Windows 95 OSR2 and later) are accessible in Windows Server 2003.

Reserved sector: این داده دوبایتی نشان دهنده آدرس شروع fat می‌باشد. در واقع این آدرس محل شروع جدول تخصیص حافظه یا memory allocation table می‌باشد که مهمترین بخش فضای حافظه بوده و در ادامه توضیح داده خواهد شد.

با توجه به اینکه بیشتر مواقع آدرس سکتور اصلی صفر است بطور معمول آدرس شروع fat نیز همیشه 36 می‌باشد. البته این آدرس در مواردی که شرکت راه‌انداز نیاز به قرار دادن اطلاعاتی دور از دسترس کاربران خود داشته باشد فرق خواهد کرد اما اگر مموری برای کار با ویندوز طراحی شده باشد این آدرس همیشه 36 است.

0x0E	2 bytes	24 00	**Reserved Sectors.** The number of sectors that precede the start of the first FAT, including the boot sector.

Number of FATs: این داده یک بایتی بوده و تعداد fat های موجود در مموری را نشان می‌دهد و به طور معمول حاوی عدد 2 بوده که نشانگر دو fat در مموری است که همانطور که قبلا گفته شد fat دوم برای بازیابی اطلاعات است که به دلایل نا معلومی از آن استفاده نمی‌شود. روش کار به صورتی است که فایل‌ها و فولدرها هم در fat اول و هم در fat دوم تخصیص و آدرس دهی می‌شوند اما در هنگام پاک کردن یا فرمت سریع یا quick format فقط fat اول پاک شده و fat دوم تا زمانیکه فایل یا فولدر جدیدی ایجاد نشود اطلاعات مربوط به داده‌های قبلی را دارد پس می‌توان با رجوع به fat

دوم اطلاعات از دست رفته را بازگرداند اما همانطور که گفته شد این روش استفاده نمی‌شود و در هنگام پاک کردن یا فرمت سریع هر دو fat پاک می‌شوند.

0x10	1 byte	02	**Number of FATs**. The number of copies of the FAT on the volume. The value of this field is always 2.

Root entries و small sectors: این دو داده که هر کدام دوبایت می‌باشند در fat32 همیشه عدد صفر را در خود قرار ذخیره کرده‌اند.

0x11	2 bytes	00 00	**Root Entries (FAT12/FAT16 only)**. For FAT32 volumes, this field must be set to zero.
0x13	2 bytes	00 00	**Small Sectors (FAT12/FAT16 only)**. For FAT32 volumes, this field must be set to zero.

*: تعدادی داده بعد از داده‌های بالا قرار دارد که به کار ما در این پروژه نیامده و یا در مورد انواع دیگر حافظه می‌باشد و توضیح آن به پیچیدگی پروژه می‌افزاید.

0x15	1 byte	F8	**Media Descriptor**. Provides information about the media being used. A value of 0xF8 indicates a hard disk and 0xF0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 disks and are not used in Windows Server 2003.
0x16	2 bytes	00 00	**Sectors Per FAT (FAT12/FAT16 only)**. For FAT32 volumes, this field must be set to zero.
0x18	2 bytes	3F 00	**Sectors Per Track**. Contains the "sectors per track" geometry value for disks that use INT 13h. The volume is broken down into tracks by multiple heads and cylinders.
0x1A	2 bytes	FF 00	**Number of Heads**. Contains the "count of heads" geometry value for disks that use INT 13h. For example, on a 1.44-MB, 3.5-inch floppy disk this value is 2.
0x1C	4 bytes	3F 00 00 00	**Hidden Sectors**. The number of sectors on the volume before the boot sector. This value is used during the boot sequence to calculate the absolute offset to the root directory and data areas. This field is generally only relevant for media that are visible on interrupt 13h. It must always be zero on media that are not partitioned.

Largs sectors: یکی از مهمترین داده‌های موجود در فایل سیستم که چهار بایتی و تعداد سکتورهای موجود در حافظه را نمایش می‌دهد. آدرس آن بایت 33 به بعد بوده و با توجه به جدول نمونه حافظه 128 مگابایتی این اعداد ملاحظه می‌شوند: 00 197 03 00 که با محاسبه آن خواهیم داشت 0x3C500 یا 247040 که ضرب در 512 فضای کل حافظه 126484480 بایت خواهد شد که بر 1024 خواهیم داشت 123520 کیلوبایت و بر 1024 خواهیم داشت 120.625 مگابایت.

E-mail: ali.tarroosheh@gmail.com

0x20	4 bytes	1D 91 11 01	**Large Sectors**. Contains the total number of sectors in the FAT32 volume.

Sectors per FAT: این داده چهار بایتی تعداد سکتور اشغالی هر fat یا همان memory allocation table را نشان می‌دهد و به زبان دیگر تنها و تنها راه پیدا کردن شماره سکتور root directory استفاده از این داده می‌باشد. روش محاسبه به صورت زیر است

Rootdirsectornumber=reservedsector + (sectorperfat * numberoffats)

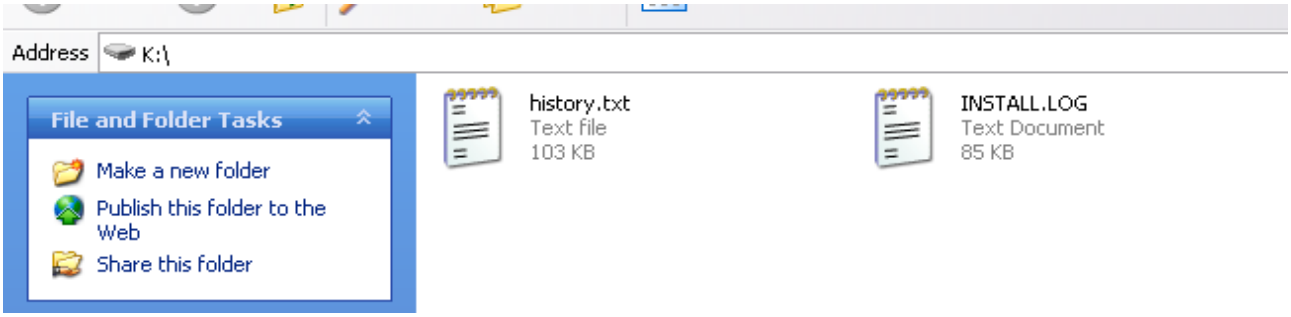
که در حافظه 128 مگابایتی شاخه اصلی برابر است با:

Rootdirsectornumber = 36 + (2 * 958) = 1952

این آدرس سکتور شاخه اصلی است. حال با سیستم طراحی شده domino این آدرس را می-خوانیم و با محتوی مموری کارت مقایسه می‌کنیم:

```
INSTALL LOG ]úc/9/9ju_9_—P_HISTORY TXT _rúc/9/9Yb04X_š_
```

که در محیط ویندوز داریم:



*. در این سکتور داده‌هایی در مورد محتوی شاخه وجود دارد که در ادامه توضیح خواهیم داد. اما به طور کلی اسم فایل و فولدرهای موجود قابل رویت است.

0x24	4 bytes	2A 22 00 00	**Sectors Per FAT (FAT32 only)**. The number of sectors occupied by each FAT on the volume. The computer uses this number and the number of FATs and reserved sectors (described in this table) to determine where the root directory begins. The computer can also determine where the user data area of the volume begins based on the number of entries in the root directory.

*. دو داده دوبایتی بدون کاربرد بعد از sector per FAT قرار دارد.

0x28	2 bytes	00 00	Not used by Windows Server 2003.
0x2A	2 bytes	00 00	**File System Version (FAT32 only)**. The high byte is the major revision number; the low byte is the minor revision number. This field supports the ability to extend the FAT32 media type in the future with concern for old FAT32 drivers mounting the volume. Both bytes are zero in Windows Server 2003, Windows 2000, and Windows Me and earlier.

Root cluster number: از آنجایی که شاخه اصلی ممکن است فایل و فولدرهای زیادی را در خود جای دهد به همین دلیل این شاخه شامل چند کلاستر است که در این داده این تعداد مشخص

E-mail: ali.tarroosheh@gmail.com

می‌شود. طول این داده چهار بایت است و به قول مایکروسافت بطور معمول شامل عدد 2 است یعنی شاخه اصلی دو کلاستر است.

0x2C	4 bytes	02 00 00 00	**Root Cluster Number (FAT32 only)**. The cluster number of the first cluster of the root directory. This value is typically, but not always, 2.

*. بعد از این داده چند بایت قرار دارد که کاربردی در پروژه ندارد.

0x30	2 bytes	01 00	**File System Information Sector Number (FAT32 only)**. The sector number of the File System Information (FSINFO) structure in the reserved area of the FAT32 volume. The value is typically 1. A copy of the FSINFO structure is kept in the Backup Boot Sector, but it is not kept up-to-date.
0x32	2 bytes	06 00	**Backup Boot Sector (FAT32 only)**. A value other than zero specifies the sector number in the reserved area of the volume where a copy of the boot sector is stored. The value of this field is typically 6. No other value is recommended.
0x34	12 bytes	00 00 00 00 00 00 00 00 00 00 00 00	**Reserved (FAT32 only)**. Reserved space for future expansion. The value of this field must always be zero.

BPB: بعد از داده‌های بالا چند بایت برای انحصاری کردن مموری تخصیص داده شده که مربوط به نام و شماره مشخصه مموری و ... است.

Extended BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Sample Value	Field Name and Definition
0x40	1 byte	80	**Physical Drive Number**. Related to the BIOS physical drive number. Floppy drives are identified as 0x00 and physical hard disks are identified as 0x80, regardless of the number of physical disk drives. Typically, this value is set prior to issuing an INT 13h BIOS call to specify the device to access. This value is only relevant if the device is a boot device.
0x41	1 byte	00	**Reserved**. FAT32 volumes are always set to zero.
0x42	1 byte	29	**Extended Boot Signature**. A field that must have the value 0x28 or 0x29 to be recognized by Windows Server 2003.
0x43	4 bytes	F1 9E 5E 5E	**Volume Serial Number**. A random serial number that is created when a volume is formatted and that helps to distinguish between disks.
0x47	11 bytes	NO NAME	**Volume Label**. A field that was once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x52	8 bytes	FAT32	**System ID**. A text field with a value of FAT32.

بقیه داده‌ها مربوط به داده‌های قابل اجراست که در این پروژه بکار نیامده و فقط در همه مموری‌ها ثابت بوده و در هنگام فرمت کردن باید در محل مناسب نوشته شوند.

ساختار Root directory:

E-mail: ali.taroosheh@gmail.com

این داده‌ها مربوط به یکی از فایل‌هاست که خوانده شده:

```
point1:(I);73
point2:(N);78
point3:(S);83
point4:(T);84
point5:(A);65
point6:(L);76
point7:(L);76
point8:( );32
point9:(L);76
point10:(O);79
point11:(G);71
point12:( );32
point13:();0
point14:(]);93
point15:(ú);250
point16:(c);99
point17:(/);47
point18:(9);57
point19:(/);47
point20:(9);57
point21:();0
point22:();0
point23:(j);106
point24:(u);117
point25:(_);4
point26:(9);57
point27:(_);3
point28:();0
point29:(—);151
point30:(P);80
point31:(_);1
point32:();0
```

ساختار داده‌ها و اطلاعات موجود در این سکتورها همانند بقیه دایرکتوری‌ها بوده و تفاوتی بین fat32 و fat16 در این مورد نیست. در این محیط به‌ازای هر فایل یا فولدر 32 بایت فضا اشغال شده و به ترتیب یازده بایت اولیه آن مربوط به اسم و پسوند فایل یا فولدر می‌باشد. فرمت این اسم به صورت 8.3 می‌باشد یعنی هشت کارکتر اسم و سه کارکتر پسوند. در صورتیکه طول اسم کمتر باشد فضای موجود با کد اسکی space یا 0x20 پر می‌شود و نکته قابل توجه این است که کارکتر ’.‘ در این فضا وجود ندارد و سیستمی که در حال کار با این فضا است باید بجای ’.‘ به تعداد فضای خالی کد 0x20 را قرار دهد و در هنگام خواندن نیز سه بایت آخر پسوند در نظر گرفته شده و فضاهای خالی حذف بقیه اسم در نظر گرفته می‌شوند.

بایت بعدی معروف به بایت هویت یا Attribute بوده که در آن اطلاعاتی از قبیل فایل/ فولدر بودن یا hidden/file system/VID/read only/Archive مشخص می‌شود. جدول زیر توضیح این داده‌هاست:

E-mail: ali.taroosheh@gmail.com

Attrib Bit	Function	LFN	Comment
0 (LSB)	Read Only	1	Should not allow writing
1	Hidden	1	Should not show in dir listing
2	System	1	File is operating system
3	Volume ID	1	Filename is Volume ID
4	Directory	x	Is a subdirectory (32-byte records)
5	Archive	x	Has been changed since last backup
6	Ununsed	0	Should be zero
7 (MSB)	Ununsed	0	Should be zero

به طور مثال اگر فضای 32 بایتی موجود مربوط به فولدر باشد بیت چهارم این داده یک می‌باشد. یا در داده‌های مربوط به مموری point(12):32 یعنی بیت پنجم یک است پس فایل Archive می‌باشد.

چند داده بعد مربوط به آخرین زمان دسترسی و زمان تولید فایل می‌باشد:

Creation time and date	5 bytes	Time and date file was created.
Last access date	2 bytes	Date file was last accessed.
Last modification time and date	4 bytes	Time and date file was last modified.

بایت 27 و 28 مربوط به آدرس کلاستر شروع فایل یا فولدر می‌باشد. این آدرس مربوط به فضای عمومی نبوده بلکه آدرس شروع در جدول تخصیص حافظه را نشان می‌دهد که توضیح خواهیم داد.

از بایت 29 تا 32 نیز ظرفیت فایل را نمایش می‌دهد که در مثال فوق داریم 151 80 01 00 که ظرفیت فایل 86167 بایت خواهد بود.

نکته: برای پاک کردن فایل یا فولدر کافی است بجای کارکتر اول اسم آن عدد 0xE5 را قرار داد. در این صورت سیستمی که فایل یا فولدری با شروع 0xE5 را مشاهده کرد آن را نادیده می‌گیرد. درصورتیکه بایت اول هر 32 بایت صفر باشد سیستم باید آنرا آخر فولدر در نظر بگیرد و برای ایجاد فایل یا فولدر جدید از اینجا شروع کند.

E-mail: ali.taroosheh@gmail.com

جدول تخصیص حافظه

همانطور که قبلا توضیح داده شد فضای FAT یا جدول تخصیص حافظه بطور معمول در یک مموری دو کپی برابر هم داشته که به FAT اول و دوم معروف است. آدرس FAT دوم را می‌توان با محاسبات زیر بدست آورد:

SECONDFATADD = RESERVEDSECTOR + SECTORPERFAT

اما چون این دو فضا معادل همدیگر می‌باشند FAT اول را توضیح می‌دهیم.

برای درک بهتر موضوع سکتور اول MEMORY ALLOCATION TABLE را که سکتور 36 است بوسیله DOMINO بررسی می‌کنیم:

point1:(ø);248
point2:(ÿ);255
point3:(ÿ);255
point4:( );15
point5:(ÿ);255
point6:(ÿ);255
point7:(ÿ);255
point8:(ÿ);255
point9:(ÿ);255
point10:(ÿ);255
point11:(ÿ);255
point12:( );15
point13:( );4
point14:();0
point15:();0
point16:();0
point17:( );5
point18:();0
point19:();0
point20:();0
point21:( );6
point22:();0
point23:();0
point24:();0
point25:( );7
point26:();0
point27:();0
point28:();0
point29:( );8
point30:();0
point31:();0
point32:();0
point33:( );9
point34:();0
point35:();0
point36:();0
point37:();10
point38:();0
point39:();0
point40:();0
point41:( );11
point42:();0
point43:();0
point44:();0
point45:();12
point46:();0
point47:();0
point48:();0
point49:( );13
point50:();0
point51:();0
point52:();0
point53:();14
point54:();0
point55:();0
point56:();0
point57:( );15
point58:();0
point59:();0
point60:();0
point61:( );16
point62:();0
point63:();0
point64:();0
point65:( );17
point66:();0
point67:();0
point68:();0
point69:( );18
point70:();0
point71:();0
point72:();0
point73:( );19
point74:();0
point75:();0
point76:();0
point77:( );20
point78:();0
point79:();0
point80:();0
point81:( );21
point82:();0
point83:();0
point84:();0
point85:( );22
point86:();0
point87:();0
point88:();0
point89:( );23
point90:();0
point91:();0
point92:();0
point93:( );24
point94:();0
point95:();0
point96:();0
point97:( );25
point98:();0
point99:();0
point100:();0
point101:( );26
point102:();0
point103:();0
point104:();0
point105:( );27
point106:();0
point107:();0
point108:();0
point109:( );28
point110:();0
point111:();0
point112:();0
point113:( );29
point114:();0
point115:();0
point116:();0
point117:(-);30
point118:();0
point119:();0
point120:();0
point121:();31
point122:();0
point123:();0
point124:();0
point125:( );32
point126:();0
point127:();0
point128:();0
point129:(!);33
point130:();0
point131:();0
point132:();0
point133:(");34
point134:();0
point135:();0
point136:();0
point137:(#);35
point138:();0
point139:();0
point140:();0
point141:($);36
point142:();0
point143:();0
point144:();0
point145:(%);37
point146:();0
point147:();0
point148:();0
point149:(&);38
point150:();0
point151:();0
point152:();0
point153:(');39
point154:();0
point155:();0
point156:();0
point157:(();40
point158:();0
point159:();0
point160:();0
point161:());41
point162:();0
point163:();0
point164:();0
point165:(*);42
point166:();0
point167:();0
point168:();0
point169:(+);43
point170:();0
point171:();0
point172:();0
point173:(,);44
point174:();0
point175:();0
point176:();0
point177:(-);45
point178:();0
point179:();0
point180:();0
point181:(.);46
point182:();0

E-mail: ali.tarroosheh@gmail.com

point183:();0
point184:();0
point185:(/);47
point186:();0
point187:();0
point188:();0
point189:(0);48
point190:();0
point191:();0
point192:();0
point193:(1);49
point194:();0
point195:();0
point196:();0
point197:(2);50
point198:();0
point199:();0
point200:();0
point201:(3);51
point202:();0
point203:();0
point204:();0
point205:(4);52
point206:();0
point207:();0
point208:();0
point209:(5);53
point210:();0
point211:();0
point212:();0
point213:(6);54
point214:();0
point215:();0
point216:();0
point217:(7);55
point218:();0
point219:();0
point220:();0
point221:(8);56
point222:();0
point223:();0
point224:();0
point225:(9);57
point226:();0
point227:();0
point228:();0
point229:(:);58
point230:();0
point231:();0
point232:();0
point233:(;);59
point234:();0
point235:();0
point236:();0
point237:(<);60
point238:();0
point239:();0
point240:();0
point241:(=);61
point242:();0
point243:();0
point244:();0
point245:(>);62
point246:();0
point247:();0
point248:();0
point249:(?);63
point250:();0
point251:();0
point252:();0
point253:(@);64
point254:();0
point255:();0
point256:();0
point257:(A);65
point258:();0
point259:();0
point260:();0
point261:(B);66
point262:();0
point263:();0
point264:();0
point265:(C);67
point266:();0
point267:();0
point268:();0
point269:(D);68
point270:();0
point271:();0
point272:();0
point273:(E);69
point274:();0
point275:();0
point276:();0
point277:(F);70
point278:();0
point279:();0
point280:();0
point281:(G);71
point282:();0
point283:();0
point284:();0
point285:(H);72
point286:();0
point287:();0
point288:();0
point289:(I);73
point290:();0
point291:();0
point292:();0
point293:(J);74
point294:();0
point295:();0
point296:();0
point297:(K);75
point298:();0
point299:();0
point300:();0
point301:(L);76
point302:();0
point303:();0
point304:();0
point305:(M);77
point306:();0
point307:();0
point308:();0
point309:(N);78
point310:();0
point311:();0
point312:();0
point313:(O);79
point314:();0
point315:();0
point316:();0
point317:(P);80
point318:();0
point319:();0
point320:();0
point321:(Q);81
point322:();0
point323:();0
point324:();0
point325:(R);82
point326:();0
point327:();0
point328:();0
point329:(S);83
point330:();0
point331:();0
point332:();0
point333:(T);84
point334:();0
point335:();0
point336:();0
point337:(U);85
point338:();0
point339:();0
point340:();0
point341:(V);86
point342:();0
point343:();0
point344:();0
point345:(W);87
point346:();0
point347:();0
point348:();0
point349:(ÿ);255
point350:(ÿ);255
point351:(ÿ);255
point352:( );15
point353:(Y);89
point354:();0
point355:();0
point356:();0
point357:(Z);90
point358:();0
point359:();0
point360:();0
point361:([);91
point362:();0
point363:();0
point364:();0
point365:(\);92
point366:();0
point367:();0
point368:();0
point369:(]);93
point370:();0
point371:();0
point372:();0
point373:(^);94
point374:();0
point375:();0
point376:();0
point377:(_);95
point378:();0
point379:();0
point380:();0
point381:(`);96
point382:();0
point383:();0
point384:();0
point385:(a);97
point386:();0
point387:();0
point388:();0
point389:(b);98
point390:();0
point391:();0
point392:();0
point393:(c);99
point394:();0
point395:();0
point396:();0
point397:(d);100
point398:();0
point399:();0
point400:();0
point401:(e);101
point402:();0
point403:();0
point404:();0
point405:(f);102
point406:();0
point407:();0
point408:();0
point409:(g);103
point410:();0
point411:();0
point412:();0
point413:(h);104
point414:();0
point415:();0
point416:();0
point417:(i);105
point418:();0
point419:();0
point420:();0
point421:(j);106
point422:();0
point423:();0
point424:();0
point425:(k);107
point426:();0
point427:();0
point428:();0
point429:(l);108
point430:();0

E-mail: ali.tarroosheh@gmail.com

point431:();0
point432:();0
point433:(m);109
point434:();0
point435:();0
point436:();0
point437:(n);110
point438:();0
point439:();0
point440:();0
point441:(o);111
point442:();0
point443:();0
point444:();0
point445:(p);112
point446:();0
point447:();0
point448:();0
point449:(q);113
point450:();0
point451:();0

point452:();0
point453:(r);114
point454:();0
point455:();0
point456:();0
point457:(s);115
point458:();0
point459:();0
point460:();0
point461:(t);116
point462:();0
point463:();0
point464:();0
point465:(u);117
point466:();0
point467:();0
point468:();0
point469:(v);118
point470:();0
point471:();0
point472:();0

point473:(w);119
point474:();0
point475:();0
point476:();0
point477:(x);120
point478:();0
point479:();0
point480:();0
point481:(y);121
point482:();0
point483:();0
point484:();0
point485:(z);122
point486:();0
point487:();0
point488:();0
point489:({);123
point490:();0
point491:();0
point492:();0
point493:(|);124

point494:();0
point495:();0
point496:();0
point497:(});125
point498:();0
point499:();0
point500:();0
point501:(~);126
point502:();0
point503:();0
point504:();0
point505:( );127
point506:();0
point507:();0
point508:();0
point509:(€);128
point510:();0
point511:();0
point512:();0

همانطور که گفته شد در FAT32 آدرس دهی 32 بیتی می‌باشد که در این مثال به خوبی قابل رویت است و تفاوت FAT32 با FAT16 تنها در طول آدرس‌هاست. در FAT16 طول آدرس دهی 16 بیت یا دو بایت است.

برای راحتی کار بهتر است هر چهار بایت داده را معرف یک کلاستر بشناسیم. در این صورت در FAT32 دوازده بایت اولیه یا دو کلاستر اولیه همیشه اعداد موجود در مثال فوق را خواهند داشت که مربوط به تعداد کلاستر به ازای شاخه اصلی می‌شود. در مثال صفحه 23 دیدیم که در 32 بایت داده مربوط به فایل بایت 27 و 28 عدد 3 را در خود نگه داشته‌اند که با رابطه زیر می‌توان آدرس شروع فایل یا فولدر را بدست آورد.(عدد موجود در بایت 27 و 28 را add در نظر بگیرید)

Starttsector = dirsectornumber + ((add-2)*sectorpercluster)

با توجه به رابطه فایل مورد نظر با add=3 و dirsectornumber=1952 و sectorpercluster=2 از آدرس 1952 شروع می‌شود.

پس تا اینجا آدرس شروع یک فایل بدست آمد اما ادامه فایل در کدام سکتور یا کلاستر وجود دارد. اینجا وجود جودل تخصیص حافظه ضرورت می‌یابد. در مثال add=3 بود پس در جدول تخصیص درایه سوم یعنی بایت 13 تا 16 مربوط به کلاستر ادامه فایل است. حال در این مثال در درایه سوم عدد 4 نوشته شده پس کلاستر ادامه فایل با توجه به رابطه فوق 1956 خواهد بود.

با توجه به توضیحات بالا می‌توان دریافت که با خواندن هر کلاستر باید سیستم به جدول رجوع کند و آدرس کلاستر ادامه را بیابد و این روند تا جایی ادامه دارد که در درایه مربوط در جدول عدد 15 255 255 255 یا 0x0FFFFFFF نوشته شود که این داده یعنی این کلاستر کلاستر آخر فایل است که در این مثال برای فایل اول point349 تا point352 موید این موضوع است.

E-mail: ali.tarroosheh@gmail.com

به راحتی می‌توان دریافت که تمام محتوی کلاستر آخر مربوط به فایل نمی‌باشد که این موضوع را باید از حجم فایل منهای حجم خوانده شده پیدا کرد که به این موضوع اصطلاحا EOF یا end of file گویند و با صفر شدن آن فایل به پایان خود رسیده است و اطلاعات بعدی موجود در کلاستر آخر مربوط به این فایل نیست.

نکته بسیار مهم در جدول تخصیص اینستکه اگر فایلی را ایجاد کرده و بعد از اینکه چندین فایل یا فولدر را ایجاد کنید سپس دوباره فایل اول را ویرایش کرده و حجم آن افزایش یابد ادامه فایل دیگر مانند مثال بالا در ادامه کلاستر قبل نخواهد بود (چون این فضا قبلا اشغال شده) بلکه آدرس اولین کلاستر خالی بعد در آن قرار خواهد گرفت. در مثال زیر یک نمونه فایل ویرایش شده را در جدول تخصیص ملاحظه می‌کنید. ابتدا دو فایل install.log و history.txt ایجاد شده و سپس install.log دوباره ویرایش شده:

point1:(ø);248
point2:(ÿ);255
point3:(ÿ);255
point4:( );15
point5:(ÿ);255
point6:(ÿ);255
point7:(ÿ);255
point8:(ÿ);255
point9:(ÿ);255
point10:(ÿ);255
point11:(ÿ);255
point12:( );15
point13:(&);38
point14:();0
point15:();0
point16:();0
point17:( );5
point18:();0
point19:();0
point20:();0
point21:( );6
point22:();0
point23:();0
point24:();0
point25:( );7
point26:();0
point27:();0
point28:();0
point29:( );8
point30:();0
point31:();0
point32:();0
point33:( );9
point34:();0
point35:();0
point36:();0
point37:();10
point38:();0
point39:();0
point40:();0
point41:(
);11
point42:();0
point43:();0
point44:();0
point45:();12
point46:();0
point47:();0
point48:();0
point49:(
);13
point50:();0
point51:();0
point52:();0
point53:();14
point54:();0
point55:();0
point56:();0
point57:( );15
point58:();0
point59:();0
point60:();0
point61:( );16
point62:();0
point63:();0
point64:();0
point65:( );17
point66:();0
point67:();0
point68:();0
point69:( );18
point70:();0
point71:();0
point72:();0
point73:( );19
point74:();0
point75:();0
point76:();0
point77:( );20
point78:();0
point79:();0
point80:();0
point81:( );21
point82:();0
point83:();0
point84:();0
point85:( );22
point86:();0
point87:();0
point88:();0
point89:( );23
point90:();0
point91:();0
point92:();0
point93:( );24
point94:();0
point95:();0
point96:();0
point97:( );25
point98:();0
point99:();0
point100:();0
point101:( );26
point102:();0
point103:();0
point104:();0
point105:( );27
point106:();0
point107:();0
point108:();0
point109:( );28
point110:();0
point111:();0
point112:();0
point113:( );29
point114:();0
point115:();0
point116:();0
point117:(-);30
point118:();0
point119:();0
point120:();0
point121:();31
point122:();0
point123:();0
point124:();0
point125:( );32
point126:();0
point127:();0
point128:();0
point129:(!);33
point130:();0
point131:();0
point132:();0
point133:(");34
point134:();0
point135:();0
point136:();0
point137:(#);35
point138:();0
point139:();0
point140:();0
point141:($);36
point142:();0
point143:();0
point144:();0
point145:(%);37
point146:();0
point147:();0
point148:();0
point149:(ÿ);255
point150:(ÿ);255
point151:(ÿ);255
point152:( );15
point153:(');39
point154:();0
point155:();0
point156:();0
point157:(();40
point158:();0
point159:();0
point160:();0
point161:());41
point162:();0
point163:();0
point164:();0
point165:(*);42
point166:();0

E-mail: ali.taroosheh@gmail.com

point167:();0
point168:();0
point169:(+);43
point170:();0
point171:();0
point172:();0
point173:(,);44
point174:();0
point175:();0
point176:();0
point177:(ÿ);255
point178:(ÿ);255
point179:(ÿ);255
point180:( );15
point181:();0
point182:();0
point183:();0
point184:();0
point185:();0
point186:();0
point187:();0
point188:();0
point189:();0
point190:();0
point191:();0
point192:();0
point193:();0
point194:();0
point195:();0
point196:();0
point197:();0
point198:();0
point199:();0
point200:();0
point201:();0
point202:();0
point203:();0
point204:();0
point205:();0
point206:();0
point207:();0
point208:();0
point209:();0
point210:();0
point211:();0
point212:();0
point213:();0
point214:();0
point215:();0
point216:();0
point217:();0
point218:();0
point219:();0
point220:();0
point221:();0
point222:();0
point223:();0
point224:();0
point225:();0
point226:();0
point227:();0
point228:();0
point229:();0
point230:();0
point231:();0
point232:();0
point233:();0
point234:();0
point235:();0
point236:();0
point237:();0
point238:();0
point239:();0
point240:();0
point241:();0
point242:();0
point243:();0
point244:();0
point245:();0
point246:();0
point247:();0
point248:();0
point249:();0
point250:();0
point251:();0
point252:();0
point253:();0
point254:();0
point255:();0
point256:();0
point257:();0
point258:();0
point259:();0
point260:();0
point261:();0
point262:();0
point263:();0
point264:();0
point265:();0
point266:();0
point267:();0
point268:();0
point269:();0
point270:();0
point271:();0
point272:();0
point273:();0
point274:();0
point275:();0
point276:();0
point277:();0
point278:();0
point279:();0
point280:();0
point281:();0
point282:();0
point283:();0
point284:();0
point285:();0
point286:();0
point287:();0
point288:();0
point289:();0
point290:();0
point291:();0
point292:();0
point293:();0
point294:();0
point295:();0
point296:();0
point297:();0
point298:();0
point299:();0
point300:();0
point301:();0
point302:();0
point303:();0
point304:();0
point305:();0
point306:();0
point307:();0
point308:();0
point309:();0
point310:();0
point311:();0
point312:();0
point313:();0
point314:();0
point315:();0
point316:();0
point317:();0
point318:();0
point319:();0
point320:();0
point321:();0
point322:();0
point323:();0
point324:();0
point325:();0
point326:();0
point327:();0
point328:();0
point329:();0
point330:();0
point331:();0
point332:();0
point333:();0
point334:();0
point335:();0
point336:();0
point337:();0
point338:();0
point339:();0
point340:();0
point341:();0
point342:();0
point343:();0
point344:();0
point345:();0
point346:();0
point347:();0
point348:();0
point349:();0
point350:();0
point351:();0
point352:();0
point353:();0
point354:();0
point355:();0
point356:();0
point357:();0
point358:();0
point359:();0
point360:();0
point361:();0
point362:();0
point363:();0
point364:();0
point365:();0
point366:();0
point367:();0
point368:();0
point369:();0
point370:();0
point371:();0
point372:();0
point373:();0
point374:();0
point375:();0
point376:();0
point377:();0
point378:();0
point379:();0
point380:();0
point381:();0
point382:();0
point383:();0
point384:();0
point385:();0
point386:();0
point387:();0
point388:();0
point389:();0
point390:();0
point391:();0
point392:();0
point393:();0
point394:();0
point395:();0
point396:();0
point397:();0
point398:();0
point399:();0
point400:();0
point401:();0
point402:();0
point403:();0
point404:();0
point405:();0
point406:();0
point407:();0
point408:();0
point409:();0
point410:();0
point411:();0
point412:();0
point413:();0
point414:();0

E-mail: ali.tarroosheh@gmail.com

point415:();0	point440:();0	point465:();0	point490:();0
point416:();0	point441:();0	point466:();0	point491:();0
point417:();0	point442:();0	point467:();0	point492:();0
point418:();0	point443:();0	point468:();0	point493:();0
point419:();0	point444:();0	point469:();0	point494:();0
point420:();0	point445:();0	point470:();0	point495:();0
point421:();0	point446:();0	point471:();0	point496:();0
point422:();0	point447:();0	point472:();0	point497:();0
point423:();0	point448:();0	point473:();0	point498:();0
point424:();0	point449:();0	point474:();0	point499:();0
point425:();0	point450:();0	point475:();0	point500:();0
point426:();0	point451:();0	point476:();0	point501:();0
point427:();0	point452:();0	point477:();0	point502:();0
point428:();0	point453:();0	point478:();0	point503:();0
point429:();0	point454:();0	point479:();0	point504:();0
point430:();0	point455:();0	point480:();0	point505:();0
point431:();0	point456:();0	point481:();0	point506:();0
point432:();0	point457:();0	point482:();0	point507:();0
point433:();0	point458:();0	point483:();0	point508:();0
point434:();0	point459:();0	point484:();0	point509:();0
point435:();0	point460:();0	point485:();0	point510:();0
point436:();0	point461:();0	point486:();0	point511:();0
point437:();0	point462:();0	point487:();0	point512:();0
point438:();0	point463:();0	point488:();0	
point439:();0	point464:();0	point489:();0	

در بایتهای 13 تا 16 عدد 38 نوشته شده است یعنی سیستم بعد از add=3 یا سکتور شروع 1954 و خواندن یک کلاستر باید برای ادامه به آدرس add=38 یا سکتور 2024 مراجعه کند. با کمی محاسبه در این مثال درایه 38 برابر با point153 تا point156 خواهد بود که کلاستر ادامه را add=39 معرفی کرده و ...

نکته: برای فولدر در جدول داده 0X0FFFFFFF را قرار می‌دهند.

در ادامه بدلیل مشابهت‌های FAT16 به FAT32 از توضیح آن خودداری می‌کنیم.

E-mail: ali.taroosheh@gmail.com

بیسکام و توابع کتابخانه‌ای AVR-DOS

یکی از امکانات پرقدرت و پرکاربرد کامپایلر BASCOM برای میکروکنترلرهای AVR کتابخانه AVR-DOS می‌باشد که شامل دو فایل MMC.LBX و AVR-DOS.LBX می‌باشد و سرعت برنامه‌نویسی را بالا برده و پیچیدگی بدست گرفتن FAT و مموری کارت را کاهش می‌دهد البته این کتابخانه شامل همه کاربردهای FAT نبوده و فقط برای ایجاد فایل و فولدر کارایی دارد و هیچ دیدی از FAT به کاربر خود نمی‌دهد. این کتابخانه توسط Josef Franz Vögel تهیه شده و در سایت http://members.aon.at/voegel در دسترس می‌باشد. از امکانات لایبری AVR-DOS می‌توان به موارد زیر اشاره کرد:

INITFILESYSTEM: این تابع در ابتدای کار مموری کارت و میکروکنترلر باید فراخوانی شود تا مواردی که قبلا ارائه شد از قبیل ROOT DIRECTORY و SECTOR PER FAT و ... از مموری خوانده شود. ساختار:

```
bErrorCode = INITFILESYSTEM (bPartitionNumber)
```

OPEN: این تابع یک فایل با اسم مورد نظر ایجاد می‌کند و به آن یک شماره اختصاص می‌دهد. نوع فایل می‌تواند BINARY برای نوشتن و خواندن بایت به بایت خارج از کد اسکی باشد، INPUT برای فقط خواندنی، OUTPUT برای فقط نوشتنی و APPEND برای فایلی که قبلا موجود بوده و هم می‌توان در آن نوشت و هم خواند. ساختار:

```
OPEN file FOR MODE as #channel
```

که در آن FILE اسم، MODE همان انواع نام برده بالا و CHANNEL یک شماره مخصوص این فایل است. مثال:

```
OPEN “ALI.TXT” FOR BINARY AS #1
```

در اینجا یک فایل با نام ALI.TXT از نوع دسترسی باینری با شماره 1 تولید شد. از این پس برای کار با این فایل با شماره آن کار می‌کنیم.

CLOSE: بستن یک فایل ک قبلا باز شده. این دستور به طور اتوماتیک محتوی فایل را ذخیره نیز می‌کند. ساختار:

```
CLOSE #CHANNEL
```

FLUSH: برای ذخیره سازی فایل از این دستور استفاده می‌شود.

PRINT: برای نوشتن یک خط رشته در قالب اسکی از این دستور استفاده می‌شود. ساختار:

```
PRINT #CHANNEL,(STRING)
```

مثال:

```
PRINT #1,”MY NAME IS ALI TAROOSHEH”
```

LINE INPUT: برای خواندن یک خط در قالب اسکی از این تابع استفاده می‌شود. ساختار:

```
LINEINPUT #bFileNumber, sLineText
```


E-mail: ali.taroosheh@gmail.com

مثال:

```
LINE INPUT #1,VAR
```

در اینجا متغییر VAR از نوع رشته است.

LOC: این تابع محل فعلی خواندن یا نوشتن را ارائه می‌دهد. ساختار:

```
lLastReadWritten = LOC (#bFileNumber)
```

نکته: تمام خروجی و ورودی‌هایی که با محل و ظرفیت سرو کار دارند از جنس LONG یا متغییر چهار بایتی می‌باشند.

LOF: حجم فایل را اراوه می‌دهد. ساختار:

```
lFileLength = LOF (#bFileNumber)
```

EOF: وضعیت پایان فایل را نشان می‌دهد. در صورتی که 255 باشد یعنی آخر فایل.

```
bFileEOFStatus = EOF(#bFileNumber)
```

FREEFILE: این تابع برای پیدا کردن یک شماره آزاد برای ایجاد فایل جدید می‌باشد. با مثال زیر کاربرد و ساختار مشخص است:

```
FF=FREEFILE()
OPEN “ALI.TXT” FOR BINARY AS #FF
```

که نوع متغییر FF از جنس بایت است و بجای اینکه شماره را مستقیما ذکر کنیم متغییر FF ملاک خواهد بود. به اینصورت:

```
PRINT #FF,”MY NAME IS ALI TAROOSHEH”
```

FILEATTR: با ساختار زیر نوع فایل باز شده را نشان می‌دهد:

```
bFileAttribut = FILEATTR(bFileNumber)
```

که عدد بازگشتی از جدول زیر مشخص می‌شود:

Return value	Open mode
1	INPUT
2	OUTPUT
8	APPEND
32	BINARY

BSAVE: یک داده را در فایل مورد نظر ذخیره می‌کند. این تابع کاربرد چندانی ندارد.

BLOAD: این تابع مقداری را از فایل می‌خواند. این تابع نیز کاربرد زیادی ندارد.

KILL: برای پاک کردن یک فایل از این دستور استفاده می‌شود:

```
KILL sFileName
```

مثال:

```
KILL “ALI.TXT”
```

SEEK: برای جابجایی در فایل از این دستور استفاده می‌شود. این تابع مانند کرسرها در ویندوز محل نوشتن یا خواندن را در فایل جابجا می‌کند. ساختار:

E-mail: ali.taroosheh@gmail.com

```
NextReadWrite = SEEK (#bFileNumber)
SEEk #bFileNumber, NewPos
```

ساختار اول برای پیدا کردن محل و ساختار دوم برای جابجایی می‌باشد.

DISKFREE: این تابع فضای خالی مموری را بدست می‌آورد. اسن تابع بعلت اینکه وابسته به میزان فضای اشغالی مموری می‌باشد زمان متغییری را لازم دارد ولی در هر حالت زمان طولانی را نیاز دارد. ساختار:

```
lFreeSize = DISKFREE()
```

DISKSIZE: این تابع جحم کلی مموری را برمی‌گرداند. ساختار:

```
Lsize=disksize()
```

GET: این تابع بخش مورد نظری از فایل را می‌خواند. ساختار:

```
GET #FILENUMBER,VAR,POSITION,LENGTH
```

متغییر VAR برای خروجی تابع و POSITION برای محل ابتدای خواندن و LENGTH تعداد بایت‌های مورد نیاز برای خواندن می‌باشد. مثال:

```
GET #1,S,234,50
```

PUT: این دستور برعکس تابع بالا برای نوشتن در فایل می‌باشد. ساختار:

```
PUT #FILENUMBER,VAR,POSITION,LENGTH
```

FILEDATE: تاریخ تولید فایل را برمی‌گرداند. ساختار:

```
sDate = FILEDATE ()
sDate = FILEDATE (file)
```

FILETIME: ساعت تولید فایل را برمی‌گرداند. ساختار:

```
sTime = FILETIME ()
sTime = FILETIME (file)
```

FILEDATETIME: برای بازگرداندن ساعت و تاریخ تولید فایل از این تابع استفاده می‌شود. ساختار:

```
Var = FILEDATETIME ()
Var = FILEDATETIME (file)
```

DIR: برای بدست آوردن محتوی مسیر موجود از این تابع استفاده می‌شود که ساختار آن را در یک مثال در زیر مشاهده می‌کنید:

```
S = Dir("*.*")
WhileLen(s)> 0 ' if there was a file found
  Print S ;" ";Filedate();" ";Filetime();" ";Filelen()
  S = Dir()
Wend
```

در این مثال در خط اول نوع جستجو را مشخص کرده و تا زمانی که طول خروجی تابع بزرگتر از صفر است حلقه ادامه میابد.

E-mail: ali.tarroosheh@gmail.com

FILELEN: ایت تابع طول فایل را بازمی‌گرداند. ساختار:

```
lSize = FILELEN ()
lSize = FILELEN (file)
```

WRITE: برای نوشتن داده در فایل استفاده می‌شود که کاربرد چندانی ندارد.

INPUT: برای خواندن یک خط در قالب اسکی از این تابع استفاده می‌شود. ساختار:

```
INPUT #CHANNEL,VAR
```

MKDIR: برای تولید یک فولدر از این تابع استفاده می‌شود. ساختار:

```
MKDIR VAR
```

CHDIR: برای تغییر مسیر از این تابع استفاده می‌شود. ساختار:

```
CHDIR VAR
```

برای بازگشت به فولدر قبل:

```
CHDIR “.”
```

برای بازگشت به شاخه اصلی:

```
CHDIR “\”
```

RMDIR: برای پاک کردن یک فولدر خالی از این تابع استفاده می‌شود. ساختار:

```
RMDIR VAR
```

اما در ادامه شرکت MCSELEC در بیسکام چندین لایبری دیگر برای کار با انواع حافظه‌ها مانند CF، HDD و MMC/SD ارائه داده که توابع کتابخانه MMC.LBX که برای MMC/SD است را بررسی می‌کنیم:

DriveCheck: این تابع مموری را چک می‌کند تا از آمادگی آن اطلاع حاصل کند. ساختار:

```
bErrorCode = DRIVECHECK()
```

اگر خروجی صفر باشد درایو آماده کار است.

DriveReset: برای شروع به کار همانطور که در ابتدای این مطلب ارائه شد مموری ابتدا باید ریست شود که این تابع همین کار را انجام می‌دهد. ساختار:

```
bErrorCode = DRIVERESET()
```

DriveInit: این دستور نیز برای INIT کردن مموری است که قبلا توضیح داده شد. ساختار:

```
bErrorCode = DRIVEINIT()
```

DriveWriteSector: برای نوشتن در یک سکتور از این تابع استفاده می‌شود. ساختار:

```
bErrorCode = DRIVEWRITESECTOR(wSRAMPointer, lSectorNumber)
```

DriveReadSector: برای خواندن یک سکتور از این تابع استفاده می‌شود. ساختار:

```
bErrorCode = DRIVEREADSECTOR(wSRAMPointer, lSectorNumber)
```

این لایبری دو فایل دیگر ارائه شده که برای تنظیمات کاربری ایجاد شده یکی config_mmc.bas که شامل تنظیم سرعت ارتباط SPI و پایه‌های متصل به مموری و سخت-افزاری/نرم‌افزاری بودن آن و ... است و دیگری CONFIG_AVR-DOS.bas که تنظیمات مربوط

به فضای اشغالی هر فایل در SRAM و تعدادی متغییر که مواردی از قبیل SECTORPERCLUSTER و ROOTFIRSTSECTOR و ... در آن قرار دارد و یک متغییر برای نمایش نوع خطای رخ داده.

E-mail: ali.tarroosheh@gmail.com

Domino مهره‌های به هم پیوسته با وظیفه محدود

در اینجا بدلیل گستردگی برنامه ابتدا آن را به دو بخش تقسیم کرده و به ذکر توضیح هر کدام بطور جداگانه می‌پردازیم. تقسیمات:

1. بخش مدیریت موری یا memory card manager: این بخش در ادامه توضیحات این متن ایجاد شد تا به کار بر خود اجازه انجام کارهای زیر را انجام دهد:

الف: ایجاد فایل دلخواه و نوشتن/خواندن و به طور کلی همه اعمال ویرایش فایل.

ب: ایجاد فولدر/تغییر نام فایل و فولدر/پاک کردن فایل و فولدر.

ج: تغییر مسیر در شاخه‌های مختلف/ بازگشت به شاخه اصلی/ بازگشت به شاخه قبلی.

د: بررسی وضعیت مموری از نظر حجم کل و فضای خالی و موادری از قبیل نوع FAT و...

ه: توانایی فرمت کردن مموری از 16 مگا بایت تا 4 گیگا بایت طبق دو استاندارد fat32 و fat16 به صورت سریع و کلی.

و: توانایی بازیابی فایل و فولدرهای پاک شده یا Disk recovery.

2. بخش اجرایی که به کاربر امکان این را می‌دهد تا کل سیستم را بوسیله یک یا چند فایل اجرایی ویژه در دست بگیرد. روش کار بگونه‌ای است که کاربر سیستم بجای اینکه برنامه میکروکنترلر را عوض کند تا به خواسته خود برسد کافی است در برنامه‌ای که در ویندوز طراحی شده است خواسته خود را پیاده کند. این برنامه تحت ویندوز و netframework. بوده و بسیار شبیه به یک اسمبلر میکروکنترلر عمل می‌کند و یک فایل متنی را تبدیل به فایل اجرایی سیستم می‌کند (System Control File) در این صورت کل فضای مموری کارت با حداکثر فضای 4 گیگابایت را می‌توان به نوعی فضای Rom میکروکنترلر یا Program memory خواند. اما چون سرعت سیستم در صورتی که یک فایل حجیم در حال اجرا باشد بسیار کم می‌شود به همین دلیل طرح دومینو برای رفع مشکل و بالا بردن کارایی مطرح می‌شود. در این طرح فایل‌هایی با حجم محدود که بتوان همه آن را در فضای sram قرار داد ایجاد شده و فایل‌ها بسته به شرایط مانند مهره‌های دومینو پشت سر هم عمل خود را انجام می‌دهند(باز می‌شوند اجرا می‌شوند و فایل دیگری را باز و اجرا می‌کنند و خودشان بسته می‌شوند). در واقع هر فایل یک تابع بوده و طی روند برنامه فایل یا تابع دیگری را اجرا می‌کند به همین دلیل امکان بازگشت به فایل قبل نیز بوجود آمد. این قسمت توانایی انجام موارد زیر را دارد:

الف: اعمال جابجایی داده.

ب: اعمال ریاضی و منطقی.

ج: شرطی با پرش.

E-mail: ali.taroosheh@gmail.com

د: پرش با بازگشت و پرش بدون بازگشت.

ه: باز کردن فایل برای ذخیر و بازیابی اطلاعات (با این امکان می‌توان داده‌ها را لاگ کرد یا از آن بعنوان پایگاه داده استفاده کرد البته در اینجا محدودیت حجم وجود ندارد).

و: ذخیره وضعیت فعلی سیستم در یک فایل(این امکان مانند ذخیره محل پرش در استک یک میکروکنترلر است که با دستور بازگشت، محل قبلی از استک بازیابی می‌شود در اینجا در یک فایل ذخیره می‌شود تا برای بازگشت دوباره، اطلاعات موجود در فایل مذکور در دستور کار سیستم قرار بگیرد).

ز: اجرای فایل دیگر. در این حالت فایل فعلی بسته شده و فایل دیگر باز می‌شود.

ح: از دیگر امکانات این قسمت اجرای توابع گسترده‌تر از حیطه دستورات اسمبلی بوده که در اختیار کاربر قرار گرفته است.

توضیحات بخش اول: میکروکنترلر Atmega64 در فرکانس کاری 14.7456 مگاهرتز با واسط کاربری سریال یا RS232 که بصورت هوشمند بوده و از باود ریت استاندارد 1200 بیت بر ثانیه تا 614400 بیت بر ثانیه را پشتیبانی می‌کند. در این بخش سرعت ارتباط میکروکنترلر با مموری کارت 8mbps تنظیم و شامل دو پین دستکانی error برای بروز خطا و busy برای نشان دادن وضعیت اشغال می‌باشد. سیستم بعد از پیکره‌بندی‌های اولیه از قبیل تنظیم RTC یا ساعت زمان واقعی، تنظیم سرعت RS232، پیکره بندی دو لایبری نام برده شده، وضعیت ECHO در RS232 و بررسی وجود مموری در سوکت، بررسی سلامت و صحت عملکر FILE SYSTEM، و بدست آوردن حجم مموری از روی رجیستر داخلی مموری کارت بنام CSD (این عمل در مواقعی که مموری کارت آسیب دیده باشد حیاطی می‌باشد زیرا در صورت آسیب دیدن FILE SYSTEM سیستم از حجم مموری مطلع نمی‌باشد) و موارد دیگر آماده دریافت دستور از کاربر می‌باشد در حین این تنظیمات پیغام‌هایی مبنی بر وضعیت عملکرد روی پورت سریال ارسال می‌شود.

نکته: در این بخش از آنجایی که سیستم قابلیت فرمت مموری داراست در صورتی که مموری آسیب دیده باشد الزامی بود که ظرفیت مموری از روی file system تعیین نشود زیرا به احتمال زیاد قابل شناسایی نخواهد بود. تنها را رفع مشکل استفاده از رجیستر داخلی مموری به نام CSD می‌باشد. این رجیستر با طول 128 بیت حاوی اطلاعات مهم زیر می‌باشد:

E-mail: ali.taroosheh@gmail.com

Table 3-11 CSD Register Fields

Field	Width	Cell Type	CSD Slice	CSD Value	CSD Code	Description
CSD_STRUCTURE	2	R	[127:126]	1.0	0	CSD structure
---	6	R	[125:120]	---	000000b	Reserved
TAAC	8	R	[119:112]	1.5 msec	00100110	Data read access time-1
NSAC	8	R	[111:104]	0	00000000b	Data read access time-2 in CLK cycles (NSAC*100)
TRANS_SPEED	8	R	[103:96]	Default 25MHz High-speed 50MHz	0110010 01011010	Max. data transfer rate
CCC	12	R	[95:84]	All (inc. WP, lock/unlock)	5F5	Card command classes
READ_BL_LEN	4	R	[83:80]	2G Up to 1G	Ah 9h	Max. read data block length
READ_BL_PARTIAL	1	R	[79:79]	Yes	1b	Partial blocks for read allowed
WRITE_BLK_MISALIGN	1	R	[78:78]	No	0b	Write block misalignment
READ_BLK_MISALIGN	1	R	[77:77]	No	0b	Read block misalignment
DSR_IMP	1	R	[76:76]	No	0b	DSR implemented
---	2	R	[75:74]	---	00b	Reserved
C_SIZE	12	R	[73:62]	2 GB 1 GB 512 MB 256 MB 128 MB 64 MB 32 MB 16 MB	F24h F22h F1Eh F13h F03h EDFh 74Bh 383h	Device size
VDD_R_CURR_MIN	3	R	[61:59]	100 mA	111b	Max. read current @VDD min.
VDD_R_CURR_MAX	3	R	[58:56]	80 mA	110b	Max. read current @VDD

E-mail: ali.tarroosheh@gmail.com

Field	Width	Cell Type	CSD Slice	CSD Value	CSD Code	Description
						max.
VDD_W_CURR_MIN	3	R	[55:53]	100 mA	111b	Max. write current @VDD min.
VDD_W_CURR_MAX	3	R	[52:50]	80 mA	110b	Max. write current @VDD max.
C_SIZE_MULT	3	R	[49:47]	2G=2048 1G=1024 512=512 256=256 128=128 64=64 32=32 16=16	0x07 0x07 0x06 0x05 0x04 0x03 0x03 0x03	Device size multiplier
ERASE_BLK_EN	1	R	[46:56	Yes	1b	Erase single block enable
SECTOR_SIZE	7	R	[45:39]	32 blocks	0011111b	Erase sector size
WP_GRP_SIZE	7	R	[38:32]	128 sectors	1111111b	Write protect group size
WP_GRP_ENABLE	1	R	{31:31]	Yes	1b	Write protect group enable
Reserved	2	R	[30:29]	---	0b	Reserved for MMC compatibility
R2W_FACTOR	3	R	[28:26]	x16	0100b	Write speed factor
WRITE_BL_LEN	4	R	[25:22]	2G Up to 1G	Ah 9h	Max. write data block length
WRITE_BL_PARTIAL	1	R	[21:21]	No	0	Partial blocks for write allowed
---	5	R	[20:16]	---	00000b	Reserved
FILE_FORMAT_GRP	1	R/W (1)	[15:15]	0	0b	File format group

E-mail: ali.tarroosheh@gmail.com

COPY	1	R/W (1)	[14:14]	Not original	1b	Copy flag (OTP)
PERM_ WRITE_ PROTECT	1	R/W (1)	[13:13]	Not protected	0b	Permanent write protection
TMP_WRITE_ PROTECT	1	R/W	[12:12]	Not protected	0b	Temporary write protection
FILE_ FORMAT	2	R/W (1)	[11:10]	HD w/partition	00b	File format
Reserved	2	2	R/W	[9:8]	---	Reserved
CRC	7	R/W	[7:1]	---	CRC7	CRC
---	1	---	[0:0]	---	1b	Not used, always "1"

با توجه به این جدول و روابط موجود که از طرف تولید کننده ارائه شده می‌توان ظرفیت را بدست آورد. روابط زیر به نقل برگه اطلاعات مموری کارت‌های SD از شرکت SanDisk می‌باشد:

- **C_SIZE (Device Size)**—computes the card capacity. The memory capacity of the card is computed from the entries C_SIZE, C_SIZE_MULT and READ_BL_LEN as follows:

  memory capacity = BLOCKNR * BLOCK_LEN

  Where:

  $$BLOCKNR = (C\_SIZE+1) * MULT$$

  $$MULT = 2^{C\_SIZE\_MULT+2} \qquad (C\_SIZE\_MULT < 8)$$

  $$BLOCK\_LEN = 2^{READ\_BL\_LEN} \qquad (READ\_BL\_LEN < 12)$$

  Therefore, the maximum capacity that can be coded is 4096*512*2048=4 GB. For example, a 4-MB card with BLOCK_LEN = 512 can be coded with C_SIZE_MULT = 0 and C_SIZE = 2047.

*. ProductManualSDCardv2.2.PDF صفحه 37.

نکته: در مبحث بازیابی داده این نکته دارای اهمیت زیادی است که فایل پاک شده در جدول تخصیص حافظه از بین رفته و هر درایه از این جدول که برای فایل مورد نظر باشد با صفر پر می‌شود، و از آنجایی که در هنگام تولید فایل جدید سیستم باید در جدول اولین فضای خالی یا صفر را برای شروع فایل در نظر بگیرد و ادامه فایل را در فضاهای خالی بعد قرار دهد در صورتیکه فایلی پاک شود و فایل دیگری ایجاد شود دیگر نمی‌توان فایل از دست رفته را بازگرداند. این حالت در شرایطی که فایلی دوبار یا بیشتر ویرایش شده باشد نیز صادق است زیرا درهنگام بازیابی فیلهای پاک شده در هم ادغام می‌شوند. با توجه به این توضیحات سیستم مورد نظر در ریکاوری بدنبال جای خالی می‌گردد و آنرا پر می‌کند پس رعایت این نکته هم مهم است که در هنگان بازیابی فایل/فولدر ها باید به ترتیب ارائه شده بازیابی شوند. (تمام این محدودیت‌ها بدلیل اینستکه در این سیستم یک فضای مموری یا پارتیشن

وجود دارد و اگر فایل جدیدی را برای ریکاوری باز کنیم همه داده‌ها از بین می‌روند پس به ناچار خود فایل را در جای خودش بازیابی می‌کنیم)

دستورات قابل پشتیبانی به شرح زیر می‌باشند:

DF: فضای خالی مموری.

DS: حجم کلی مموری.

DIR: نمایش محتوی شاخه فعلی.

SF: نمایش سایز فایل مورد نظر.

DEL: پاک کردن فایل.

REN: تغییر نام فایل.

CD: تغییر مسیر.

MD: تولید فولدر خالی.

RD: پاک کردن فولدر خالی.

OP: باز کردن فایل.

T/D: تغییر ساعت و تاریخ.

TIME: نمایش ساعت و تاریخ.

CD: بازگشت به شاخه اصلی.

CD..: بازگشت به شاخه قبلی.

BAUD: تغییر سرعت RS232.

PROP: بررسی مشخصات مموری کارت.

RSEC: خواندن سکتور.

WSEC: نوشتن سکتور.

FORMAT: فرمت کردن مموری کارت در قالب‌های FAT32 و FAT16.

RST: راه‌اندازی مجدد سیستم با تایمر نگهبان.

SRSEC: خواندن یک مجموعه از سکتورها.

FIXBAUD: خارج کردن تنظیم سرعت از وضعیت هوشمند.

ECHO: قطع اکو در RS232. در حالت عادی اکو فعال است و هر داده‌ای که به سیستم می‌رسد دوباره بازتابانده می‌شود.

VAR: نمایش متغییرهای اصلی سیستم از جمله اطلاعات FILE SYSTEM.

RECOVERY: بازیابی داده که با این الگوریتم سیستم DOMINO بسیار سریعتر از برنامه‌های بازیابی داده موجود در ویندوز می‌باشد.

E-mail: ali.taroosheh@gmail.com

REMOVE: برای جدا کردن امن مموری از سیستم از این دستور استفاده می‌شود. با این دستور همه ارتباطات با مموری و تغذیه از آن جدا می‌شود.

COPY: برای کپی کردن یک فایل از این دستور استفاده می‌شود.

CUT: برای جابجایی یک فایل از این دستور استفاده می‌شود.

PASTE: این دستور بعد از دو دستور بالا فایل را در محل مورد نظر قرار می‌دهد.

EXE: این بخش مربوط به قسمت اجرایی سیستم بوده و یک فایل اجرایی را راه‌اندازی می‌کند.

SOUND: در این قسمت یک فایل صوتی از نوع PCM با پسوند WAV. پخش می‌شود.

E-mail: ali.tarroosheh@gmail.com

توضیحات بخش دوم: برای بخش اجرایی، یک فایل ضمیمه تعبیه شده که در پروژه با نام execute.bas می‌باشد. این فایل در ادامه متن قرار گرفته و در آن یک فایل اجرایی با ظرفیت 1024 بایت و فضای SRAM در دسترس 128 بایت، پشته به ظرفیت WORD 32 یا 64 بایت، چهار پورت ورودی/خروجی کامل و کل فضای مموری کارت در اختیار کاربر سیستم و فایل مورد نظر می‌باشد. در این سیستم یک متغییر 8 بیتی با قابلیت‌های آکومولاتور به نام A موجود می‌باشد که مانند همه سیستم‌های میکروپروسسوری همیشه در طرفین معادله حضور دارد. 128 بایت رجیستر همه کاره موجود با نام‌های R,1 تا R,128 بمنظور فضای SRAM موجود می‌باشد و یک پشته سخت افزاری به ظرفیت 32 اینتیجر یا 64 بایت برای ذخیره آدرس بازگشت از زیربرنامه تعبیه شده است. در واقع ساختار معماری یک میکروکنترلر شبیه به 8051 در این سیستم ایجاد شده با این تفاوت که توانایی توابعی بشکل ماژل را داراست. در حال حاضر هر آپکد 8 بیت طول داشته که با این شرایط می‌توان 256 دستور را ایجاد کرد که موارد ذکر شده 98 کد از این ظرفیت را اشغال کرده و 158 کد دیگر در اختیار کاربر باقی مانده که می‌توان به قابلیت‌های سیستم اضافه کرد.

نکاتی در مورد ساختار زبان برنامه‌نویسی: در مورد برنامه اجرایی که از این به بعد آنرا SCF می‌خوانیم نکات زیر الزامی است.(توضیح دستورات در ادامه داده خواهد شد)

1. اسمبلر برنامه را در قالب یک فایل نوشتاری مانند TXT. و در قالب ANSI دریافت می‌کند. و بعد از اسمبل کردن در صورتیکه خطا رخ داده باشد دو فایل با اسم فایل اصلی و با پسوند DBG. و ERR. ارائه می‌دهد. در فایل ERR. محل بروز خطا و علت آن با @@@ مشخص می‌شود. اما اگر برنامه صحیح باشد برنامه دو فایل با همان شرایط بالا ایجاد کرده یکی با پسوند DBG. و دیگری BIN. که فایل BIN. همان فایلی است که باید در مموری قرار گیرد تا اجرا شود. فایل با پسوند DBG. فایل برنامه شماست که به ساختار مورد نیاز اسمبلر تبدیل شده است. همه فایل‌های فوق در NOTEPAD قابل مشاهده می‌باشند.
2. حد واسط هر دستور و آرگومان‌های آن یک SPACE است.
3. حد واسط آرگومان‌ها با یکدیگر یک کارکتر ’,‘ می‌باشد. مثلا اگر بخواهیم عدد موجود در رجیستر 17 را در آکومولاتور قرار دهیم به این صورت می‌نویسیم:

```
MOV A,R,17
```

4. اعداد ثابت با علامت ’#‘ مشخص شوند و کارکترها باید بین ’‘‘ قرار گیرد بغیر از مواردی که در مثال‌های هر دستور مشخص شده است. مثال:

```
MOV A,#,100
MOV A,#,’@’
```

5. کارکتر ”;“ بعنوان شروع توضیحات است و کامپایلر از آنجا به بعد را کامپایل نمی‌کند. مثال:

```
MOV DDR,D,#,255; SET PORTD TO OUTPUT
```


E-mail: ali.taroosheh@gmail.com

6. کامپایلر به حروف بزرگ و کوچک حتی برای برچسب‌ها حساس نیست و می‌توان به هر حالتی برنامه را نوشت.
7. چهار پورت A,C,D,F در اختیار این قسمت است. همانطور که در AVR پورتها به صورت PIN برای ورودی و PORT برای خروجی است در اینجا نیز به این صورت است و به شکل دیگری نمی‌توان از آن استفاده کرد. بعلاوه رجیستر DDRx نیز برای کنترل PULLUP ها در اختیار کاربر قرار دار. در مثال زیر پورت C بعنوان خروجی در نظر گرفته شده و پایه-های آن یک در میان یک می‌شوند:

```
MOV DDR,C,#,255
MOV PORT,C,#,85
```

8. برای پایان برنامه نیازی به دستور خاصی نیست، در صورتی که برنامه از حلقه‌ها بیرون بیاید و به پایان برسد سیستم از حالت اجرایی در آمده و به حالت مدیریت وارد می‌شود و مانند حالت عادی مسیر مورد نظر را به منزله آماده دریافت دستور ارسال می‌کند.
9. برای تعریف ثابت از DB. یا DBS. استفاده می‌شود. DB. برای داده‌های تک کارکتری یا تک بایتی و DBS. برای داده‌های رشته. مثل:

```
.DB 12
.DB ‘A’
.DBS “MY NAME IS ALI TAROOSHEH”
```

نکته مهم اینستکه این دستورات نباید در ابتدای برنامه یا میان آن قرار بگیرد و حتی‌الامکان با یک پرش از روی این داده‌ها عبور کنید در غیر اینصورت سیستم این داده‌ها را بعنوان کد اجرایی در نظر گرفته و آنرا اجرا می‌کند. مثال:

```
JMP MAIN
.DB “WELCOME TO DOMINO TECHNOLOGY SYSTEM”
MAIN:
```

دستورات

در اینجا دستورات در دسترس را با ساختارشان در یک مثال بررسی می‌کنیم. اسم ابتدایی هر خط مربوط به برچسب آن در کد برنامه است.

Regtoa: جابجایی از R به A

```
MOV A,R,1
```

Pintoa: جابجایی از ورودی به A

```
MOV A,PIN,D
```

Dirtoa: قرار دادن داده ثابت در A

```
MOV A,#,55
```

Atoreg: جابجایی از A به R

```
MOV R,34,A
```

Pintoreg: جابجایی از ورودی به R

```
MOV R,PIN,F
```

Dirtoreg: قرار دادن ثابت در R

```
MOV R,#,255
```

Atoport: جابجایی داده از A به خروجی

```
MOV PORT,F,A
```

Regtoport: جابجایی داده از R به خروجی

```
MOV PORT,C,R,23
```

Dirtoport: قرار دادن ثابت در خروجی

```
MOV PORT,C,#,200
```

Ajmp: پرش بدون شرط و بدون بازگشت به برچسب مورد نظر

```
JMP LABEL
```

Acall: پرش بدون شرط با بازگشت به برچسب مورد نظر

```
CALL LABEL
```

Retc: بازگشت از زیر برنامه CALL

```
CALL LABEL
.
.
.
LABEL:
.
RET
```

Setb: یک کردن یک بیت

```
SET PORT,C,4
```

Clrb: صفر کردن یک بیت

```
CLR PORT,F,0
```

Dptrtoa: در سیستم یک رجیستر دو بایتی به نام DPTR در نظر گرفته شده که برای آدرس دهی فضای برنامه بکار می‌رود. محتوی اشاره شده این رجیستر با دستور زیر به A منتقل می‌شود. البته این انتقال به دو روش است، یک:

```
MOV A,@DPTR,A
```

که آدرس مورد نظر DPTR+A بوده و در حالت دوم:

```
MOV A,@DPTR,25
```

که آدرس مورد نظر برابر است با DPTR+25

Setdptr: برای مقدار دهی به DPTR با توجه به LABLE از این دستور استفاده می‌شود

E-mail: ali.tarroosheh@gmail.com

SETDPTR LABLE

Cjneareg: برای مقایسه A با R از این دستور استفاده می‌شود

CJNE A,R,21,LABLE

اگر R,21 و A با هم مخالف باشند برنامه به LABLE پرش می‌کند.

Cjneadir: این حالت برای مقایسه A و یک عدد ثابت است

CJNE A,#,12,LABLE

Cjneapin: برای مقایسه A و ورودی از این حالت استفاده می‌شود

CJNE A,PIN,D,LABLE

Djnza: حلقه کاهشی که متغییر آن A می‌باشد. در این حلقه A هر بار یک عدد کاهش می‌یابد و تا زمانیکه A بزرگتر از صفر است حلقه ادامه می‌یابد

DJNZ A,LABLE

Djnzr: این حلقه مانند حلقه بالاست ولی متغییر آن یکی از R هاست

DJNZ R,43,LABLE

Waita: این دستور در واقع یک تابع یا یک ماژول است برای ایجاد یک تاخیر برحسب ثانیه که مقدار آن در A می‌باشد

WAIT S,A

در صورتیکه بخواهیم برحسب میلی ثانیه باشد به صورت زیر عمل می‌کنیم:

WAIT MS,A

و اگر بخواهیم تاخیر بر اساس یک ثابت باشد:

WAIT S,43
WAIT MS,50

Notaa: برای NOT کردن محتوی A از این دستور استفاده می‌شود

NOT A,A

Notabit: برای NOT کردن یکی از بیت‌های A از این دستور استفاده می‌شود که آرگومان دوم شماره بیت مورد نظر است

NOT A,4

Notrr: برای NOT کردن رجیسترها این حالت استفاده می‌شود

NOT R,23,R

Notrbit: برای NOT کردن یکی از بیت‌های رجیسترها از این دستور استفاده می‌شود

NOT R,23,5

Notportddrpd: برای NOT کردن خروجی یا DDR از این حالت استفاده می‌شود

NOT PORT,C,P
NOT DDR,F,D

P نشانه پورت و D نشانه DDR

E-mail: ali.tarroosheh@gmail.com

Notportddrbit: برای NOT کردن یکی از بیت‌های خروجی یا DDR از این حالت استفاده می شود

```
NOT DDR,C,3
NOT PORT,F,2
```

Oradirect: برای OR کردن یک ثابت در A از این دستور استفاده می‌شود

```
OR A,#,120
```

Orareg: برای OR کردن یکی از رجیسترها در A از این حالت استفاده می‌شود

```
OR A,R,34
```

Oraportddr: برای OR کردن A در پورت خروجی یا DDR و قرار دادن جواب در پورت یا DDR از این حالت استفاده می‌شود

```
OR PORT,F,A
```

Oratopin: برای OR کردن ورودی در A از این حالت استفاده می‌شود

```
OR A,PIN,A
```

Andadirect: برای AND کردن یک ثابت در A از این دستور استفاده می‌شود

```
AND A,#,120
```

Andareg: برای AND کردن یکی از رجیسترها در A از این حالت استفاده می‌شود

```
AND A,R,34
```

Andaportddr: برای AND کردن A در پورت خروجی یا DDR و قرار دادن جواب در پورت یا DDR از این حالت استفاده می‌شود

```
AND PORT,F,A
```

Andatopin: برای AND کردن ورودی در A از این حالت استفاده می‌شود

```
AND A,PIN,A
```

Xoradirect: برای XOR کردن یک ثابت در A از این دستور استفاده می‌شود

```
XOR A,#,120
```

Xorareg: برای XOR کردن یکی از رجیسترها در A از این حالت استفاده می‌شود

```
XOR A,R,34
```

Xoraportddr: برای XOR کردن A در پورت خروجی یا DDR و قرار دادن جواب در پورت یا DDR از این حالت استفاده می‌شود

```
XOR PORT,F,A
```

Xoratopin: برای XOR کردن ورودی در A از این حالت استفاده می‌شود

```
XOR A,PIN,A
```

Subatodirect: کم کردن یک عدد ثابت از A

```
SUB A,#,12
```

Subatoreg: کم کردن یکی از رجیسترها از A

E-mail: ali.tarroosheh@gmail.com

SUB A,R,100

Divatodirect: تقسیم A بر یک عدد ثابت

DIV A,#,100

Divatoreg: تقسیم A بر یکی از رجیسترها

DIV A,R,128

Mulatodirect: ضرب یک عدد ثابت در A

MUL A,#,10

Mulatoreg: ضرب رجیستر در A

MUL A,R,120

Addatodirect: جمع یک عدد ثابت با A

ADD A,#,9

Addatoreg: جمع رجیستر با A

ADD A,R,1

Setdisposable: برای ذخیره سازی/بازیابی تمامی محتوی رجیسترها، ورودی/خروجی/DDR، آکومولاتور، PC، استک و اشاره‌گر آن و ... از این دستور استفاده می‌شود. برای بازیابی از آرگومان GET و برای ذخیره از آرگومان PUT استفاده می‌شود

DISPOSE GET,"ALI.DSP"

اگر نام فایل در رجیسترها باشد آرگومان‌ها یکی آدرس شروع رجیستر و دیگری طول اسم است

DISPOSE GET,R,23,10

در این مثال اسم فایل از R,23 تا R,33 قرار دارد.

نکته: طول اسم در همه حالات از 12 کارکتر بیشتر نباید باشد.

DISPOSE PUT,"ALI.DSP"

DISPOSE PUT,R,23,10

در صورتی که آرگومانی بعد از GET یا PUT قرار نگیرد فایلی با اسم فایل اجرایی و پسوند DSP. ایجاد یا بازیابی می‌شود.

DISPOSE GET

DISPOSE PUT

Dominodirect: برای باز کردن یک فایل اجرایی دیگر با دادن یک ثابت از این حالت استفاده می‌شود

DOMINO "ALI.BIN"

Dominoreg: برای باز کردن یک فایل اجرایی دیگر که نام آن در رجیسترهاست از این حالت استفاده می‌شود

DOMINO R,1,11

در این مثال R,1 آدرس شروع رشته و 11 طول رشته است که در اینجا هم طول رشته نباید از

12 بیشتر باشد.

Domino_class: این دستور العمل ترکیبی از توابع dispose و domino می‌باشد و عمل ذخیره سازی، اجرایی فایل جدید و بارگذاری setting مورد نظر روی فایل جدید اجرایی را در یک زمان انجام می‌دهد. به نوعی می‌توان گفت این دستورالعمل توانایی آن را دارد یک فایل اجرایی را با setting قبلی اجرا کرد

```
Class_domino put,”ali.dsp”-“class2.bin”-get,”class2.dsp”
```

در این ساختار آرگومان‌های اصلی با – از هم جدا می‌شوند و آرگومان اول همان dispose put می‌باشد، آرگومان دوم اسم فایل اجرایی جدید است یا همان دستور العمل domino و آرگومان سوم همان dispose get می‌باشد.

نکته: همه آرگومان‌ها مانند دستور العمل‌های اصلی ذکر شده قابلیت دریافت ثابت یا آدرس دهی از رجیستر را دارند. مثال:

```
Class_domino put-r,12,8-get,r,45,10
```

در این مثال آرگومان اول به نام فایل اجرایی فعلی با پسوند dsp. فایل ذخیره می‌کند. آرگومان دوم اسم فایلی که در فضای رجیستری از r12 به طول 8 بایت ذخیره شده را اجرا می‌کند و آرگومان سوم فایلی را که اسم آن در R45 به طول 10 بایت قرار دارد بار گذاری می‌کند. با این توضیحات هر حالت ممکن در دستورات اصلی dispose و domino را می‌توان در اینجا نیز آورد.

توضیح: این دستورالعمل در ابتدا در یک فایل مشخص وضعیت موجود را ذخیره‌سازی کرده و فایل جدید را اجرا می‌کند و تنظیمات ذخیره شده در یک فایل دیگر را روی آن قرار می‌دهد.

مثال: در اینجا یک فایل اجرایی با دستورات زیر را ایجاد می‌کنیم و اسم آنرا 1 می‌گذاریم

```
Mov ddr,a,#,255
Mov port,a,#,255
Wait s,2
Class_domino put-”2.bin”
Wait s,2
Mov port,a,#,0
```

حال فایل دومی را ایجاد می‌کنیم و اسم آنرا 2 می‌گذاریم

```
Mov ddr,a,#,255
Mov port,a,#,85
Wait s,2
Class_domino put-”1.bin”-get,”1.dsp”
```

در فایل اول پورت A خروجی شده و همه پایه‌ها set می‌شوند، بعد از دو ثانیه تاخیر وضعیت فعلی با توجه به توضیحات داده شده در فایل 1.dsp ذخیره شده و فایل دوم را اجرا می‌کند. در فایل دوم عدد 85 در پورت قرار می‌گیرد و بعد از دوثانیه فایل اول با تنظیمات موجود در 1.dsp اجرا می‌شود

E-mail: ali.taroosheh@gmail.com

این امر کاملا مشهود است زیرا بعد از بازگشت به فایل اول دو ثانیه تاخیر وجود دارد و در این دو ثانیه باید تنظیمات فایل اول ملاحظه شود یعنی مقدار پورت A عدد 255 باشد و بعد از دو ثانیه به صفر تغییر کند.

Root: دستور زیر برای بازگشت به شاخه اصلی در فایل اجرایی می‌باشد

```
CD ROOT
```

Backward: این دستور برای بازگشت به شاخه قبلی می‌باشد

```
CD BACK
```

Changdirdirect: این دستورالعمل برای تغییر مسیر به شاخه دلخواه از روی یک رشته ثابت می‌باشد

```
CD PATH,"ALI."
```

Changdirreg: این دستور مانند دستور بالا می‌باشد با این تفاوت که اسم مورد نظر در فضای رجیستری قرار دارد

```
CD PATH,R,12,11
```

Mkdirdirect: برای تولید فولدر جدید از این دستور استفاده می‌شود

```
MD PATH,"ALI."
```

Mkdirreg: برای تولید فولدر جدید با اسم موجود در فضای رجیستر از این حالت استفاده می‌شود

```
MD PATH,R,1,6
```

Jgatoreg: پرش اگر بزرگتر باشد

```
Jg a,r,23,lable
```

Jgatodirect: پرش اگر بزرگتر باشد

```
Jg  a,#,12,lable
```

Jgatopin: پرش اگر بزرگتر باشد

```
Jg a,pin,d,lable
```

Jlatoreg: پرش اگر کوچکتر باشد

```
Jl a,r,12,lable
```

Jlatodirect: پرش اگر کوچکتر باشد

```
Jl a,#,12,lable
```

Jlatopin: پرش اگر کوچکتر باشد

```
Jl a,pin,f,lable
```

Jeatoreg: پرش اگر مساوی باشد

```
Je a,r,23,lable
```

Jeatodirect: پرش اگر مساوی باشد

```
Je a,#,34,lable
```


E-mail: ali.tarroosheh@gmail.com

Jeatopin: پرش اگر مساوی باشد

Je a,pin,c,lble

Jzatoa: پرش اگر یک بیت مشخص از A صفر باشد

JZ A,3,A,LABLE

Jzpintopin: پرش اگر یک بیت از وردی‌ها صفر باشد

JZ PIN,A,4,LABLE

Jzregtoreg: پرش اگر یکی از بیت‌های رجیستر صفر باشد

JZ R,12,5,LABLE

Jnzatoa: پرش اگر یک بیت مشخص از A یک باشد

JNZ A,3,A,LABLE

Jnzpintopin: پرش اگر یک بیت از وردی‌ها یک باشد

JNZ PIN,A,4,LABLE

Jnzregtoreg: پرش اگر یکی از بیت‌های رجیستر یک باشد

JNZ R,12,5,LABLE

Rrla: چرخش آکومولاتور به راست

RRL A,3

Rrlreg: چرخش رجیستر به راست

RRL R,12,5

Rrlport: چرخش پورت به راست

RRL PORT,F,4

Rlla: چرخش آکومولاتور به چپ

RLL A,3

Rllreg: چرخش رجیستر به چپ

RLL R,23,7

Rllport: چرخش پورت به چپ

RRL PORT,C,2

Opendirect: باز کردن یک فایل برای نوشتن و خواندن داده. این دستور و دستورات دیگر مربوط به باز کردن فایل فقط برای یک فایل موثر است و باز کردن دو فایل پشت سرهم باعث بسته شدن فایل قبلی می‌شود. نکته دیگر آنکه در صورت استفاده از دستورات دومینو این فایل می‌تواند در کل مسیر دومینو باز باشد و کافی است در یکی از فایل‌ها در مسیر دومینو باز شود و همه فایل‌ها از آن استفاده کنند و کافیست در یکی از آن‌ها بسته شود، البته این فایل با خروج از وضعیت اجرایی بصورت اتوماتیک بسه می‌شود

OPEN PATH,”ALI.BIN”

Openreg: این دستور همان حالت بالاست ولی محل ذخیره سازی اسم فایل در رجیسترهاست و

E-mail: ali.tarroosheh@gmail.com

آرگومان آخر مربوط به طول اسم می‌باشد

```
OPEN R,23,10
```

Closeopen: بستن فایل مورد نظر

```
CLOSE PATH
```

Readtoa: خواندن یک بایت و قرار دادن آن در A

```
READ A
```

Writetoa: نوشتن یک بایتی A در فایل

```
WRITE A
```

Reads: خواندن یک آرایه با طول مشخص. آرگومان آخر طول داده است

```
READS R,1,100
```

Writes: نوشتن یک آرایه در فایل با طول مشخص

```
WRITES R,23,56
```

Seekputdirect: مقدار دهی به مکان نمای فایل به صورت داده ثابت. این داده می‌تواند حداکثر چهار بایتی باشد

```
SEEK PUT,#, 4281491455
```

نکته: خروج از حجم واقعی فایل امکان پذیر است و در این شرایط بروز خطا حتمی می‌باشد.پس در مقدار دهی‌ها دقت کنید که در اندازه فایل مورد نظر باشد.

Seekputreg: مقدار دهی به مکان نما از فضای رجیستری

```
SEEK PUT,R,23
```

در توضیح این حالت باید گفت که برای مقدار دهی کافیست به اولین بایت از LONG مورد نظر اشاره کرد. در این مثال رجیسترهای R,23 تا R,26 محتوی آدرس مورد نظر است

Seekadddirect: جمع کردن مکان‌نما با یک عدد ثابت با طول حداکثر چهار بایت

```
SEEK ADD,#, 4281491455
```

Seekaddreg: جمع کردن مکان‌نما با فضای رجیستری به طول چهار بایت

```
SEEK ADD,R,12
```

Seekget: خواندن مکان‌نما. در این دستور مانند دستورات بالا کافسیت به بایت ابتدایی اشاره کرد

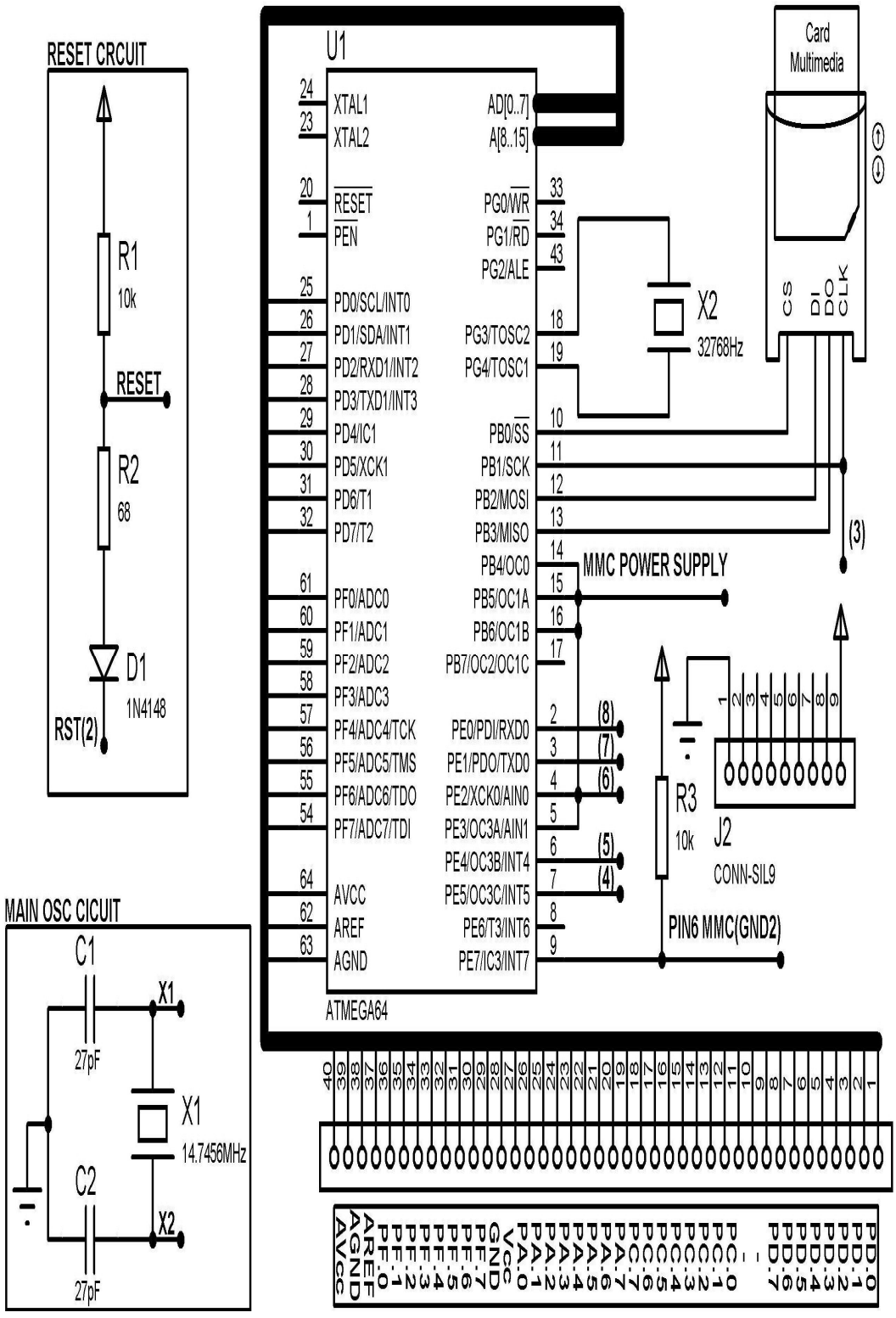
```
SEEK GET,R,45
```

در مثال بالا یک داده از نوع LONG در رجیسترهای R,45 تا R,48 قرار می‌گیرد

نکته: همانطور که قبلا گفته شد مکان‌نما در این پروژه به محل خواندن/نوشتن بعدی در فایل اشاره می‌کند.

مدار: در شکل زیر مدار طراحی شده این سیستم را ملاحظه می‌کنید.

E-mail: ali.taroosheh@gmail.com



E-mail: ali.tarroosheh@gmail.com

مبحث پایانی

همانطور که در ابتدای این گزارش ذکر شد این پروژه در طی پنج سال تدارک دیده شد و در طول این مدت بیشتر به پیدا کردن واقعیت مطلب مشغول بوده تا طراحی، به طور مثال مطالب ارائه شده از سایت مایکروسافت ناقص و در برخی موارد غلط و گمراه کننده بود، هر چند این موضوع در مورد بقیه مراجع نیز صدق می‌کرد، تنها نتیجه‌ای که می‌توان از این موضوع گرفت آنستکه شرکت‌های بزرگ تولید کننده پروتکل‌های بزرگ و کوچک حتی با منسوخ شدن آنها بازهم راضی به ارائه مطلب صحیح و پرده برداری از آن تکنولوژی نیستند. در این پروژه هیچ گونه مطلب در اینترنت یا کتابی برای شرح ساختار جدول تخصیص حافظه یافت نمی‌شود و اینجانب برای تنها این موضوع کوچک گاهاً به بررسی بیش از 10000 سکتور و مقایسه آن با 10000 سکتور دیگر نموده که اگر در مطالب فوق ملاحظه کرده باشید هر سکتور 512 بایت بوده و هر بیت آن معنای خاص و منحصر بفردی دارد و باید توجه کرد که با عدم وجود مطالب صحیح و توانایی شبیه سازی صحیح آن در محیط کامپیوتر برای کاهش هزینه و زمان اتلافی با گذشت پنج سال پروژه در جایی قرار دارد که می‌توان از اینجا به بعد آنرا گسترش داد. هر چند برخی بر این باورند که سرعت میکروکنترلرهای AVR برای اجرای دستورات کم است ولی باید در نظر داشت که این الگوریتم قابل گسترش برروی میکروکنترلرهای بوده و در حداقل کارایی از دروغگوئی‌های فنی جدا شده است.

E-mail: ali.taroosheh@gmail.com

منابع و مأخذ

ProductManualSDCardv2.2.pdf: SanDisk corporation
/How fat works: http://technet2.microsoft.com
/How ntfs works: http://technet2.microsoft.com
FAT32: http://www.pjrc.com/tech/8051/ide/fat32.html
/default cluster size: http://support.microsoft.com/kb/140365
FAT: http://www.cse.scu.edu/~tschwarz/coen152_05/Lectures/FAT.html
FAT: fatgen103.pdf
AVR-DOS: Http://Www.mcselec.com
AVR-DOS: http://members.aon.at/voegel
میکروکنترلرهای خانواده 8051 دکتر مزیدی ترجمه دکتر سپیدنام
معماری کامپیوتر موریس مانو

شماره دفتر ثبت اختراع: [redacted]
تاریخ ثبت اختراع: 1387/8/12

Domino Technology
Designe with
Ali taroosheh
Email: ali.taroosheh@gmail.com
Website: http://www.ElectroRC.blogfa.com
Website: http://www.ElectroRC-EN.blogSPOT.com