تکلیف سوم درس ریز پردازنده و زبان اسمبلی

آریا بنائی زاده 9431029

۱- در هر یک از دستورات زیر از چه مدهای آدرس‌دهی استفاده شده است؟

دستورالعمل	مد آرس‌دهی اُپرند اول (مد آدرس دهی در صورت نداشتن اُپرند)	مد آرس‌دهی اُپرند دوم (در صورت وجود)
SUB R7, R0	مستقیم توسط ثبات	مستقیم ثبات
ANDI R0, 0x40	مستقیم توسط ثبات	مستقیم داده
RJMP 0xFF	نسبی حافظه برنامه	—
IJMP	غیر مستقیم حافظه برنامه	—
RCALL 0x1000	نسبی حافظه برنامه	—
JMP 0x1000	مستقیم حافظه برنامه	—
CPC R10, R7	مستقیم توسط ثبات	مستقیم توسط ثبات
BRTC 0x400	نسبی حافظه برنامه	—
ST –X, R0	غیر مستقیم داده با پیش کاهش	مستقیم توسط ثبات
LDI R12, 0x40	مستقیم توسط ثبات	مستقیم داده
STS 0x100, R16	مستقیم حافظه داده	مستقیم توسط ثبات
STD X+0x15, R4	غیر مستقیم با جابجایی	مستقیم توسط ثبات
Mov Rd, Rr	مستقیم توسط ثبات	مستقیم توسط ثبات
ELPM R0, Z	مستقیم توسط ثبات	آدرس دهی حافظه برنامه با آدرس ثابت
IN R0, EEDR	مستقیم توسط ثبات	مستقیم I/O

۲- برنامه‌ای بنویسید که معادل اسکی نام فامیل شما را در حافظه EEPROM میکروکنترلر بنویسید.

```
; Put letters ascii code in registers

LDI r16, 0X42; B

LDI r15, 0X41; A

LDI r14, 0X4E; N

LDI r13, 0X45; E

LDI r12, 0X49; I

; Wait for completion of previous write
sbic EECR, EEWE
rjmp EEPROM_write
; Set up address (r18:r17) in address register
```

```
out EEARH, r18
out EEARL, r17
; Write data “B”
out EEDR, r16
;move to next address and Write A
inc r17
out EEARL, r17
out EEDR, r15
;move to next address and write N
inc r17
out EEARL, r17
out EEDR, r14
;move to next address and write A
inc r17
out EEARL, r17
out EEDR, r15
;move to next address and write E
inc r17
out EEARL, r17
out EEDR, r13
;move to next address and write I
inc r17
out EEARL, r17
out EEDR, r12

; Write logical one to EEMWE

sbi EECR, EEMWE
; Start eeprom write by setting EEWE
sbi EECR, EEWE
ret
```

۳- وضعیت پرچم‌ها را پس از اجرای هر یک از دستورالعمل‌های برنامه زیر مشخص نمائید. فرض کنید کلیه پرچم‌ها پس از شروع برنامه 0 هستند.

۱. LDI R0, 0x3F → 0011 1111
۲. NEG R0 → 1100 0001
۳. BST R0, 5 (5th)
۴. ADD R0, 0x3F
۵. INC R0
۶. SEI

۱) تغییر ایجاد نمی‌شود
۲) S←1 , N←1
۳) T←0
۴) Z←1 , H←1 , S←0 , N←0
۵) H←0
۶) I←1

۴- در یک زیر روال، یک بایت داده را از ثبات EEDR از آدرس 0x100 حافظه EEPROM دریافت، آنرا به ثبات R10 منتقل، نیبل‌های آنرا جابجا، بیت شماره ۴ آنرا ۱ و بیت پنجم آن را تست کنید و پیرو آن اقدامات زیر را انجام دهید:

الف- اگر نتیجه تست بیت پنجم 1 بود، مقدار نهایی R0 را در آدرس 0x05 نسبت به مقدار فعلی ثبات Z در حافظه داده ذخیره نمائید (Z=0x9A).

ب- اگر نتیجه تست بیت پنجم 0 بود، محتوای R0 را پس از یک شیفت منطقی به چپ، در عدد 0x2 ضرب و نتیجه را در دو بایت متوالی در پشته ذخیره کنید (SP=0x300)

ج- پس از ذخیره مقدار R0 در پشته، مقدار نهایی SP چقدر است؟

حل: خانه 0x100 در واقع میشود خانه 256 ام حافظه پس EEARH یک و EEARL صفر میشود

```
EEPROM_read:
Ldi r17, 0x001
Ldi r16, 0x000
; Wait for completion of previous write
sbic EECR, EEWE
rjmp EEPROM_read
out EEARH, r17
out EEARL, r16
; Start eeprom read by writing EERE
sbi EECR, EERE
; Read data from data register
in r10, EEDR
; Swap Nibbles
Swap r10
; set bit 4
sbr r10,16
;T store
Bst r10,5
ret
```

الف)

```
Brtc 2

Std Z+0x05,r0

Ret
```

ب)

```
Lsl r0

Add r0,r0

Push r0

Push r0
```

ج) 0x302

۵- برنامه‌ای به زبان اسمبلی ATMega16 بنویسید که ۱۰۰ عدد که در آدرس ARRAY در حافظه برنامه قرار گرفته‌اند را به صورت صعودی مرتب کند (فرض کنید این حافظه از پیش تعریف و مقداردهی شده است).

برای مرتب کردن از bubble sort استفاده میکنیم این الگوریتم با مقایسه دو به دوی دو عدد پشت سر هم آرایه را مرتب می کند.
کد زبان c این الگوریتم به صورت زیر است:

```
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)
       for (j = 0; j < n-i-1; j++)
           if (arr[j] > arr[j+1])
              swap(&arr[j], &arr[j+1]);
}
```

برای نوشتن کد به زبان اسمبلی ابتدا آدرس های low و high آرایه را در ZHوZL ذخیره میکنیم و دو رجیستر (یکی برای شماره عدد مورد مقایسه و دیگری برای تعداد دفعات عمل مقایسه دو به دو) را با عدد 100 لود میکنیم و شروع به آوردن اعداد از Z به رجیستر میکنیم و اعداد را باهم مقایسه میکنیم اگر عدد ها در جای درست نبودند باید جایشان را عوض کنیم و دوباره در حافظه ذخیره کنیم(سابروتین swap این کار را انجام میدهد)

```
.def i = r15 ; first loop index
.def j = r14 ; second loop index
bubbleSort:
ldi ZH, high(ARRAY << 1) ; z high definition
ldi ZL, low(ARRAY << 1) ; z low definition
Ldi secondCounter, 100
firstLoop:
Ldi j, 100
secondLoop:
;load in register for comparing
lpm R17,Z+
lpm R18,Z
mov r22,ZL
cp r17,r18 ; compare 2 consecutive numbers
brlt swapNumbers ; branch to swap function
dec i ; decrement i
brne secondLoop; loop
dec j; decrement j
brne firstLoop
rjmp bubbleSort
swapNumbers:
mov r21,r18
mov r0,r17
spm           ;      store in program memory
mov ZL,R22
mov R0,R21
spm           ;      store in program memory
ret
```