استاد: دکتر هادی مرادی	HW 7 : مبانی کامپیوتر و برنامه‌سازی	810198434 محمد هجری عبدالحسینی

سؤال 1 :

```
        .ORIG   x3000
cint    LD      R3,b15      ; R3 = b15 = x8000
        LD      R1,a        ; R1 = a
        LD      R2,b        ; R2 = b
        NOT     R4,R1       ; R4 = NOT(R1) = NOT(a)
        NOT     R5,R2       ; R5 = NOT(R2) = NOT(b)
        AND     R4,R4,R2    ; R4 = R4 AND R2 = NOT(a) AND b
        AND     R5,R5,R1    ; R5 = R5 AND R1 = NOT(b) AND a
        ADD     R4,R4,R5    ; R4 = R4 + R5 = [NOT(a) AND b] + [NOT(b) AND a]
        AND     R4,R4,R3    ; R4 = R4 AND R3
                            ; R4 = ([NOT(a) AND) b] + [NOT(b) AND a]) AND x8000
        BRzp    cmp         ; IF R4 >= 0 GO TO cmp
retA    ST      R1,r        ; r = R1 = a
        BR      leave       ; GO TO leave
cmp     NOT     R2,R2       ; R2 = NOT(R2) = NOT(b)
        ADD     R2,R2,#1    ; R2 = R2+1 = NOT(b) +1 = -b
        ADD     R2,R1,R2    ; R2 = R1 + R2 = a + NOT(b) +1 = a-b
        ST      R2,r        ; r = R2 = a-b
leave   HALT
```

سؤال 2 :

```
        .ORIG   x3000
        LEA     R0,x16
        LDR     R0,R0,#0    ;  R0 = x1000
        AND     R2,R2,#0    ; Free R2
        LEA     R2,x14
        LDR     R2,R2,#0    ; R2 = 100 (lenght of Array)
        AND     R3,R3,#0    ; Free R3
                            ; R3 is Result of SUM
SUM     LDR     R1,R0,#0    ; R1 = Array[i]
        ADD     R3,R3,R1    ; SUM = SUM + Array[i]
        ADD     R0,R0,#1    ; Next index
        ADD     R2,R2,#-1
        BRP     SUM         ; Until R2 > 0 Do SUM
        AND     R4,R4,#0    ; Free R4
                            ; R4 is Final Result

        LEA     R2,xC
        LDR     R2,R2,#0    ; R2 = -100
LOOP    ADD     R3,R3,R2    ; R3 = R3 + R2 : SUM = SUM - 100
        BRN     LEAVE
        ADD     R4,R4,#1    ; Result ++
        BR      LOOP
LEAVE   LEA     R3,x6
        LDR     R3,R3,#0    ; R3 = x2000
        STR     R4,R3,#0    ; Result stored in x2000
        HALT
Array   .FILL   x1000
L1      .FILL   x0064
L2      .FILL   xFF9C
DST     .FILL   x2000
        .END
```

سؤال 3 : خانه‌های a و b شامل اعداد دلخواه هستند .

```
.ORIG   x3000
LD      R1,a        ; R1 = a
LD      R2,b        ; R2 = b
NOT     R1,R1       ; R1 = NOT(R1) = NOT(a)
AND     R3,R1,R2    ; R3 = NOT(a) AND b
NOT     R1,R1       ; R1 = a
NOT     R2,R2       ; R2 = NOT(b)
AND     R4,R1,R2    ; R4 = a AND NOT(b)
NOT     R2,R2       ; R2 = b
NOT     R3,R3       ; R3 = NOT[NOT(a) AND b] = a OR NOT(b)
NOT     R4,R4       ; R4 = NOT[a AND NOT(b)] = NOT(a) OR b
AND     R3,R3,R4    ; R3 = [a OR NOT(b)] AND [NOT(a) OR b]
NOT     R3,R3       ; R3 = NOT([a OR NOT(b)] AND [NOT(a) OR b])
                    ; R3 = [NOT(a) AND b] OR [a AND NOT(b)]
                    ; R3 = a XOR b = a^b
                    ; R3 = R1 XOR R2 = R1^R2
AND     R4,R4,#0    ; clear R4
HALT
```

سؤال 4 :

0101 0110 1110 0000	x56E0		AND	R3,R3,#0
0101 1001 0010 0000	x5920		AND	R4,R4,#0
0001 1001 0000 0010	x1902	LOOP	ADD	R3,R3,R1
0001 1001 0011 1111	x193F		ADD	R4,R4,#-1
0000 0011 1111 1101	x03FD		BRP	LOOP
1111 0000 0010 0101	xF025		TRAP	HALT

سؤال 5 :

0101 0110 1110 0000	x56E0		AND	R3, R3, #0
0001 0110 1100 0010	x16C2		ADD	R3, R3, R2
0101 1001 0010 0000	x5920		AND	R4, R4, #0
0001 1001 0000 0010	x1902		ADD	R4, R4, R2
0101 1011 0110 0000	x5B60		AND	R5, R5, #0
0001 1011 0100 0001	x1B41		ADD	R5, R5, R1
1001 1011 0111 1111	x9B7F		NOT	R5, R5
0001 1011 0110 0001	x1B61		ADD	R5, R5, #1
0001 1001 0000 0101	x1905	LOOP	ADD	R4, R4, R5
0000 1000 0000 0011	x0803		BRN	LEAVE
0001 0110 1100 0101	x16C5		ADD	R3, R3, R5
0001 1101 1010 0001	x1DA1		ADD	R6, R6, #1
0000 1111 1111 1011	x0FFB		BRNZP	LOOP
0101 1001 0010 0000	x5920	LEAVE	AND	R4, R4, #0
0101 1011 0110 0000	x5B60		AND	R5, R5, #0
1111 0000 0010 0101	xF025		TRAP	HALT

سؤال 6 :

الف) نادرست : زیرا دستور ST ، از طریق " PC + Offset " محل ذخیره سازی را تعیین می کند.
مقدار PC که نشان دهندهٔ آدرس خط بعدی برنامه هست و مشخصه. پس ما فقط مقدار Offset را می توانیم تغییر بدیم.
که محدودهٔ آن هم از -256 تا 255 است. که این اجازه را به ما نمی دهد که مقداری را در هرجای حافظه ذخیره کنیم.

ب) نادرست : زیرا دستور BR یک دستور شرطی محسوب می شود و تنها در صورت برقرار بودن شرط روند اجرا را عوض می کند.

پ) نادرست : علت نادرستی مشابه قسمت الف می باشد. یعنی نمی توان به هرخانه از حافظه دسترسی داشت و دسترسی مان به خانه های با فاصلهٔ -256 تا 255 از PC محدود است.

ت) درست .

سؤال 7 :

x3000 0001 0100 1110 0101 ADD R2, R3, #5

ADD DR SR1 1 imm5

سؤال 8 :

در خط اول ، خانهٔ x3000 ، دستور LEA R0,#5 اجرا می‌شود . که عملکرد آن بدین صورت است که:

آدرس خانهٔ PC + offset را در R0 می‌ریزد . [R0 = PC + offset = x3001+5 = x3006]

در خط بعدی ، خانهٔ x3001 ، دستور LDR R2,R0,#1 اجرا می‌شود .

محتوای خانهٔ Base + offset را در R2 می‌ریزد . [Base + offset = x3006 + 1]

محتوای خانهٔ x3007 برابر با x600C می‌باشد . ⇐ R2 = x600C

در خط بعدی ، خانهٔ x3002 ، دستور STR R2,R0,#0 قرار دارد . که محتوای R2 را در آدرس Base + offset ذخیره می‌کند . [Base + offset = x3006 + 0] محتوای خانهٔ x3006 برابر با x600C می‌شود . اما تغییری در رجیسترها رخ نمی‌دهد .

در خط بعدی برنامه ، خانهٔ x3003 ، دستور LDI R3,#1 اجرا می‌شود . این دستور ابتدا به سراغ PC + offset می‌رود. [PC+offset=x3004+1] . و سپس به آدرسی می‌رود که در خانهٔ PC + offset وجود داشت [PC+offset=x3005] ، [محتوای خانهٔ x3005 = x3008] . آنگاه محتوای خانهٔ x3008 را در R3 ذخیره می‌کند .

[محتوای خانهٔ x3008 = x356A] ⇐ R3= x356A

سؤال 9 :

مرتب سازی stack :

```
Function to sort stack :
  sort_stack (stack) {
    if stack is not empty: {
          Remove the top item and copy in x: int x = pop(stack);
          sort remaining stack : sort_stack(stack);
          Push the top item back : sorted_insert (stack, x);
    }
  }
  sorted_insert (stack, x){
        if stack is empty OR x greater than top :{
            Push x on top of the stack : push(s,x);
            return;
        }
        if top is greater : Remove the top item and copy in temp: int temp=pop(stack);
        sorted_insert (stack, x):
        put back the top item removed earlier : push (stack, temp);
  }
```

سؤال 10 :

1- عملیات insert در لیست پیوندی بسیار ساده‌تر است زیرا در آرایه لازم است تمامی خانه‌های آرایه به جلو منتقل شود تا عملیات insert انجام شود، اما در لیست پیوندی اینکار با تغییر 2 اشاره‌گر ممکن است.

2- عملیات search در آرایه مرتب شده آسانتر از یک لیست پیوندی است. زیرا دسترسی در آرایه به مقادیر از روی index با یک عملیات صورت می‌گیرد، با روش‌هایی نظیر binary search می‌توان خیلی سریعتر جستجو را انجام داد .

3- عملیات merge در لیست پیوندی ساده‌تر است . زیرا اینکار تنها با تغییر چند اشاره‌گر صورت می‌گیرد. اما در array ممکن است این کار نیازمند به اختصاص حافظه مجدد (realloc) داشته باشد که این کار به نسبت سخت‌تر و زمانبرتر است .

4- حذف کردن در لیست پیوندی بسیار ساده‌تر است. مثل مشابه با قسمت 1 .