Bài tập LEGv8

Hãy thực hiện các yêu cầu sau cho các bài tập LEGv8 bên dưới:

- Chuyển sang C/LEGv8 tương ứng.
- Cho biết ý nghĩa đoạn code (nếu có).
- Xác định giá trị trường address của tất cả các lệnh nhảy (CBZ, CBNZ, B.COND, B) (nếu có).
- Xác định giá trị thanh ghi PC và LR sau khi thực hiện lệnh BL (nếu có).
- Xác định giá trị thanh ghi PC sau khi thực hiện lệnh BR (nếu có).

1. Remember `X0, X1` are registers used for procedure arguments / results.

Address	Instruction
10000	`FUNC: ORR  X9, XZR, XZR`
10004	`LOOP: CBZ  X1, END`
10008	`ADD  X9, X9, X0`
10012	`SUBI  X1, X1, #1`
10016	`B  LOOP`
10020	`END:  ADDI  X9, X9, #100`
10024	`ADD  X0, X9, XZR`
10028	`BR  LR`

2. Assume that the C-level integer i is held in register X10, X0 holds the C-level integer called result, and X1 holds the base address of the integer MemArray.

Address	Instruction
10000	`ORR  X10, XZR, XZR`
10004	`LOOP: LDUR  X11, [X1, #0]`
10008	`ADD  X0, X0, X11`
10012	`ADDI  X1, X1, #8`
10016	`ADDI  X10, X10, #1`
10020	`SUBIS XZR, X10, 100`
10024	`B.LT  LOOP`

3. The following code fragment processes two arrays and produces an important value in result register X0. The base addresses of the arrays are stored in X0 and X1 respectively, and their sizes are stored in X2 and X3, respectively.

Address	Instruction
10000	`FUNC: LSL  X2, X2, #2`
10004	`LSL  X3, X3, #2`
10008	`ADD  X9, XZR, XZR`
10012	`ADD  X10, XZR, XZR`
10016	`OUTER:ADD  X14, X0, X10`
10020	`LDUR  X14, [X14, #0]`
10024	`ADD  X11, XZR, XZR`

```
10028   INNER:ADD   X13, X1, X11
10032         LDUR  X13, [X13, #0]
10036         SUB   X12, X13, X14
10040         CBNZ  X12, SKIP
10044         ADDI  X9, X9, #1
10048   SKIP: ADDI  X11, X11, #4
10052         SUB   X12, X11, X3
10056         CBNZ  X12, INNER
10060         ADDI  X10, X10, #4
10064         SUB   X12, X10, X2
10068         CBNZ  X12, OUTER
10072         ORR   X0, X9, XZR
10076         BR    LR
```

4. What's the result of the program in X19 ?

Address	Instruction
0x00400030	`Main:  ADDI  X0, XZR, #10`
0x00400034	`       BL    Function`
0x00400038	`       ADD   X19, X0, XZR`
0x0040003C	`       ADDI  X8, #1      // syscall number for exit`
0x00400040	`       ADDI  X0, #0      // return 0 status`
0x00400044	`       SVC   #0          // invoke syscall to exit`
0x00400048	`Func:  SUBI  SP, SP, #32`
0x0040004C	`       STUR  X19, [SP, #24]`
0x00400050	`       STUR  X20, [SP, #16]`
0x00400054	`       STUR  LR, [SP, #8]`
0x00400058	`       STUR  X0, [SP, #0]`
0x0040005C	`       ADDI  X19, XZR, #0`
0x00400060	`       ADDI  X20, XZR, #0`
0x00400064	`Loop:  SUBS  XZR, X19, X0`
0x00400068	`       B.LT  L1`
0x0040006C	`       B     FinalFunction`
0x00400070	`L1:    ADDI  X0, X20, #0`
0x00400074	`       BL    Check`
0x00400078	`       CBZ   X0, Inc_Loop`
0x0040007C	`       ADD   X19, X19, X20`
0x00400080	`Inc_Loop:  ADDI  X20, X20, #1`
0x00400084	`       LDUR  X0, [SP, #0]`
0x00400088	`       B     Loop`
0x0040008C	`Final_Func: ADDI  X0, X19, #0`
0x00400090	`       LDUR  LR, [SP, #8]`
0x00400094	`       LDUR  X20, [SP, #16]`
0x00400098	`       LDUR  X19, [SP, #24]`
0x0040009C	`       ADDI  SP, SP, #32`
0x004000A0	`       BR    LR`
0x004000A4	`Check:  ADDI  X9, XZR, #2`
0x004000A8	`       UDIV  X10, X0, X9`
0x004000AC	`       MUL   X10, X10, X9`
0x004000B0	`       SUB   X10, X0, X10`
0x004000B4	`       CBZ   X10, L2`
0x004000B8	`       ADDI  X0, XZR, #0`
0x004000BC	`       B     Final_ Check`
0x004000C0	`L2:    ADDI  X0, XZR, #1`
0x004000C4	`Final_Check:BR    LR`

5. `X0` is procedure argument and `X1` is procedure result.

Address	Instruction
10000	`FACT: SUBI  SP, SP, #16`
10004	`STUR  LR, [SP, #8]`
10008	`STUR  X0, [SP, #0]`
10012	`SUBIS XZR, X0, #1`
10016	`B.GE  L1`
10020	`ADDI  X1, XZR, #1`
10024	`ADDI  SP, SP, #16`
10028	`BR    LR`
10032	`L1:   SUBI  X0, X0, #1`
10036	`BL    FACT`
10040	`LDUR  X0, [SP, #0]`
10044	`LDUR  LR, [SP, #8]`
10048	`ADDI  SP, SP, #16`
10052	`MUL   X1, X0, X1`
10056	`BR    LR`

6. Assume that the values of a, b, i, and j are in registers X0, X1, X10, and X11, respectively. Also, assume that register X2 holds the base address of the array D.

Instruction

`for(i=0; i<a; i++)`
`  for(j=0; j<b; j++)`
`  D[4*j] = i + j;`

7. Assume that the values of i, j, k, and m are in registers X19, X20, X21, and X22, respectively.

Instruction

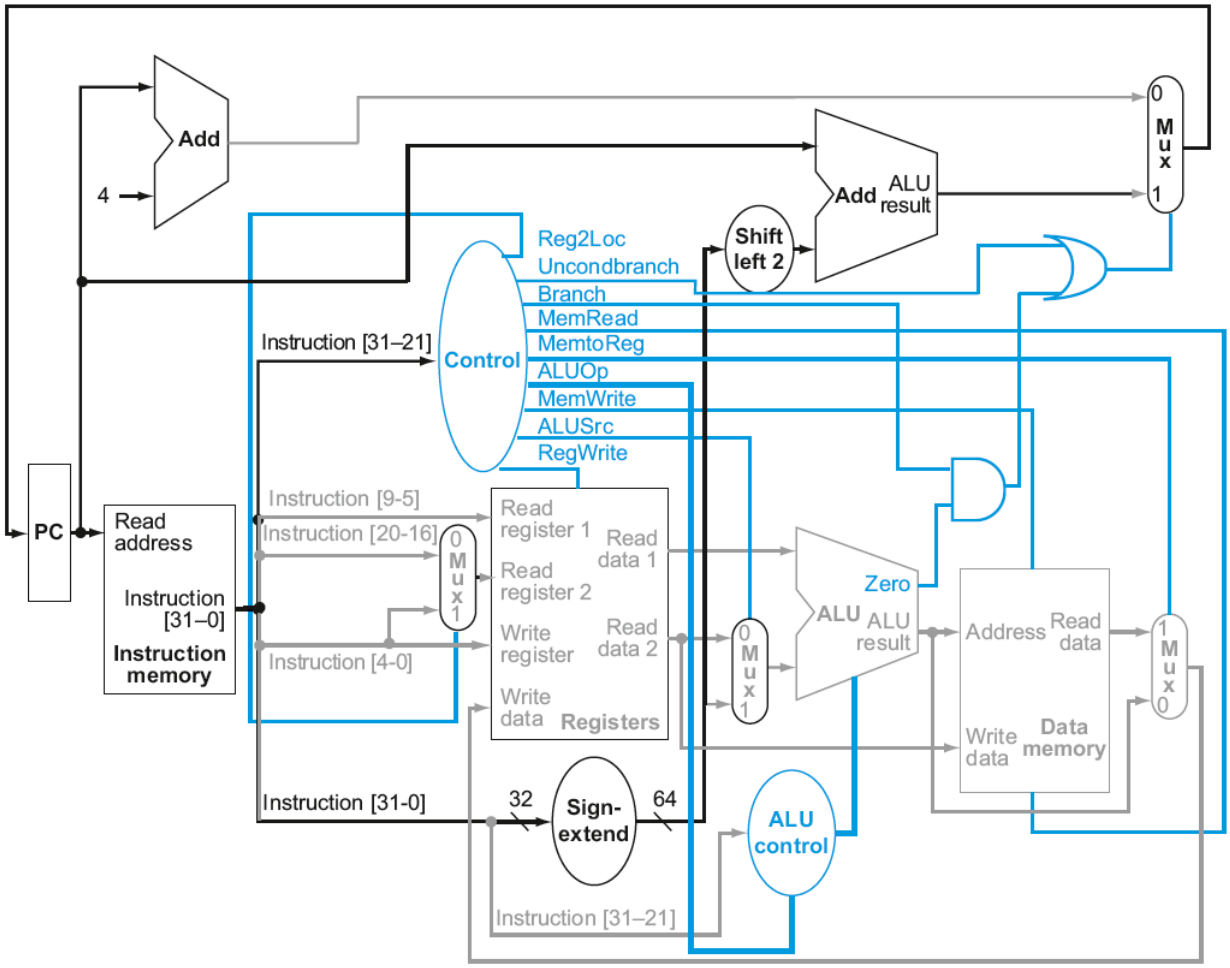
`main() {`
`  int i,j,k,m;`
`  i = mult(j,k);`
`  m = mult(i,i);`
`}`
`int mult (int mcand, int mlier) {`
`  int product;`
`  product = 0;`
`  while (mlier > 0) {`
`  product = product + mcand;`
`  mlier = mlier -1;`
`  }`
`  return product;`
`}`

8. Assume that the values of i, num are in registers X20, X21 respectively. Also, assume that register X19 holds the base address of the array.

Instruction

```
int array[10];
void main () {
      int num;
      set_array(num);
}
void set_array (int num) {
      for (int i=0; i<10; i++) {
            array[i] = compare(i,num);
            num--;
      }
}
int compare (int a, int b) {
      if (sub(b,a) >= 0)
            return b;
      else
            return a;
}
int sub (int a, int b) {
      return a-b;
}
```

9. Sử dụng sơ đồ mạch xử lý LEGv8 thu gọn như trong bài giảng để trả lời các câu hỏi sau:



9.1. Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `ADD X9, X10, X11`.

9.2. Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `LDUR X9, [X10, #0]`.

9.3. Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `STUR X9, [X10, #0]`.

9.4. Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `B L1`.

9.5. Tập lệnh LEGv8 rút gọn được sử dụng để xây dựng mạch xử lý trong bài giảng không có lệnh `EOR`. Nếu cần xây dựng thêm để xử lý lệnh `EOR` này thì cần thay đổi datapath thế nào ? Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `EOR X9, X10, X11`.

9.6. Tập lệnh LEGv8 rút gọn được sử dụng để xây dựng mạch xử lý trong bài giảng không có lệnh `CBNZ`. Nếu cần xây dựng thêm để xử lý lệnh `CBNZ` này thì cần thay đổi datapath thế nào ? Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `CBNZ X9, L1`.

9.7. Tập lệnh LEGv8 rút gọn được sử dụng để xây dựng mạch xử lý trong bài giảng không có lệnh `ADDI`. Nếu cần xây dựng thêm để xử lý lệnh `ADDI` này thì cần thay đổi datapath thế nào ? Hãy cho biết giá trị các tín hiệu điều khiển khi thực hiện lệnh `ADDI X9, X10, #1`.