

2021-2022 春夏学期 计算机组成与设计 期末考试回忆卷

By 瓜豪 & 贝拉拉贝拉 & 其他热心的98ers

Note: 允许携带一张A4, 不允许携带计算器

分布: 10道选择, 20分; 4道大题

| Name (Field size) | Field | | | | | | Comments |
|----------------------|-----------------------------|--------|--------|--------|---------------|--------|-------------------------------|
| | 7 bits | 5 bits | 5 bits | 3 bits | 5 bits | 7 bits | |
| R-type | funct7 | rs2 | rs1 | funct3 | rd | opcode | Arithmetic instruction format |
| I-type | immediate[11:0] | | rs1 | funct3 | rd | opcode | Loads & immediate arithmetic |
| S-type | immed[11:5] | rs2 | rs1 | funct3 | immed[4:0] | opcode | Stores |
| SB-type | immed[12,10:5] | rs2 | rs1 | funct3 | immed[4:1,11] | opcode | Conditional branch format |
| UJ-type | immediate[20,10:1,11,19:12] | | | | rd | opcode | Unconditional jump format |
| U-type | immediate[31:12] | | | | rd | opcode | Upper immediate format |

给出了需要用到的opcode和Fun3/7

Opcode: Beq/Bne 0x63

Fun3: Beq 0 Bne 1

选择题

1. 机器码 111111110000000000001011011100011

A. Beq x0, x0, -10 B. Beq x0, x0, -20 C. Bne x0, x0, -10 D. Bne x0, x0, -20

2. IEEE754规则化能表示的最小数。

3. PC最初为 0x20000000, 问 jal 指令所能跳转的范围

(似乎是) A. [1FF00000, 200FFFFF] B. [1FF00000, 200FFFFE] C. [1FF80000, 200FFFFE] D.

[1FF00000, 200FFFFF]

4. 提高block size可以显著影响

A. cache miss B. Compulsory miss rate C. hit time D. miss penalty

5. Exception相关

6. float加法, 0.12345×10^{-1} 与 0.12345×10^3 在对齐后, 指数位

A. both -1 B. both 3 C. both 0 D.忘记了

7. I/O相关: 根据CPU efficiency, 对3种I/O方式从高到低排序

选项就是polling, interrupt, DMA的排列组合

8. 如何获得一个比较大的立即数地址存储的内容 (原题干有些问题, 这里稍作修改)

```
A. lui x5, higher_bits  
  
   ori x5, x5, lower_bits  
  
   lw x7, 0(x5)
```

```
B. lui x5, higher_bits  
  
   or x5, x5, lower_bits  
  
   lw x7, 0(x5)
```

```
C. lui x5, higher_bits  
  
   lw x7, lower_bits(x5)
```

D忘记了。。。 Hint: 各类型立即数范围

9. 判断以下指令发生了什么：WAW(Write After Write), RAW(Read After Write), WAR(Write After Read)

```
lw x1, 0(x1)  
add x2, x1, x1  
sub x1, x2, x3
```

还有一道选择题忘记了，欢迎楼下补充

大题

CPI相关

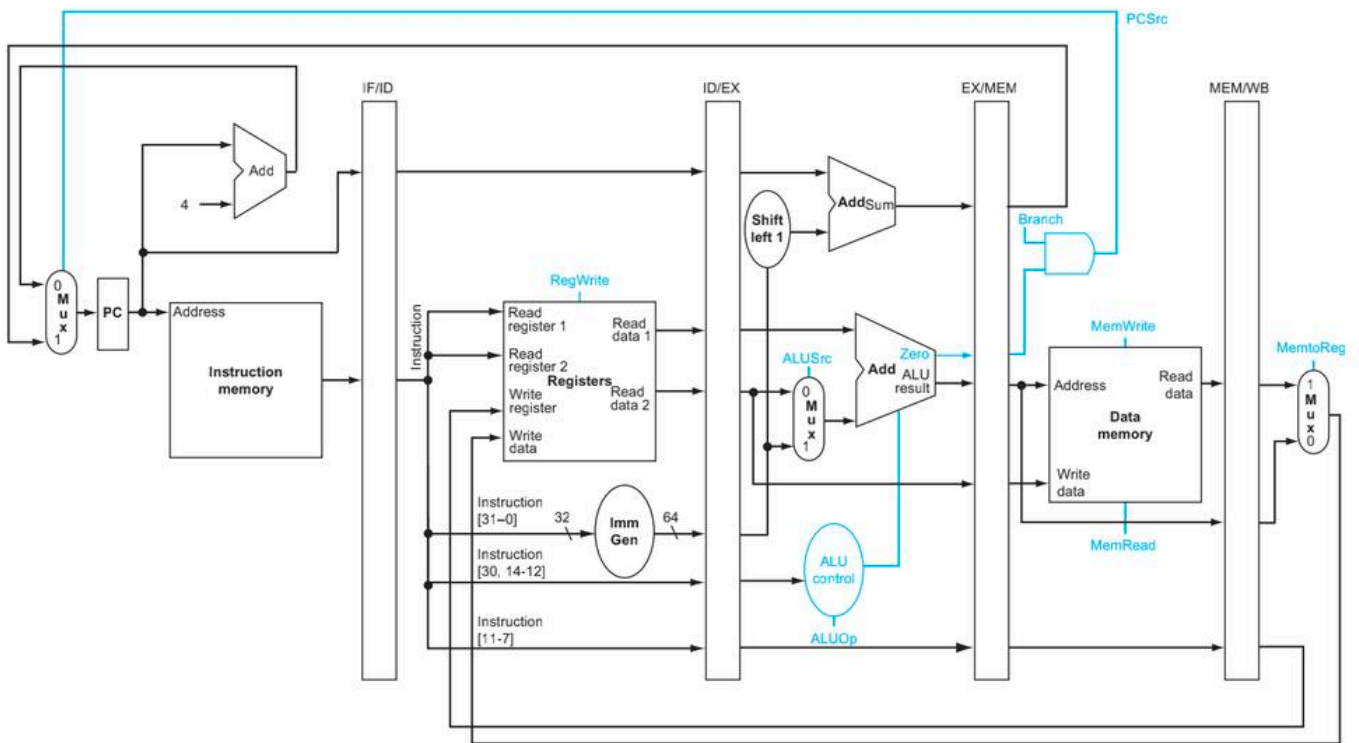
稍后我找一道类似的题目放在这里（没有找到很像的，这里我自己出一道，还是欢迎大佬补充更改数据）

| ALU/Logic | Jump/Branch | Load | Store |
|-----------|-------------|------|-------|
| 45% | 20% | 20% | 15% |

已知：5-stage 200MHz流水线CPU，具有前递，stall detector；对跳转指令采取not-taken预判，实际上45%的跳转指令not-taken，跳转的判断在4th阶段（MEM）进行；每一次从memory中fetch需要75ns；I-MEM miss rate为90%，D-MEM miss rate为98%；有25%的load指令会发生load-use hazard。

请求：该CPU的CPI值。

Datapath相关



1. SW指令，对应的CPU control信号：ALU_Src, RegWrite, MemtoReg, PCSrc
2. 给出一段代码，插入NOP（根据题目原图，无forwarding等）

```

I0  add t2, s1, sp
I1  lw  t1, 0(t1)
I2  addi t2, t1, 7
I3  add t1, s2, sp
I4  lw  t1, 0(t1)
I5  addi t1, t1, -9
I6  sub t1, t1, x2
    
```

3. 更改datapath，使其可以满足forwarding以解决算术指令的RAW竞争
4. 请在表格中填写ALU中A, B的来源（从ID/EX, EX/MEM, MEM/WB中进行选择）

| Inst. | ALU_Src_A | ALU_Src_B |
|-------|-----------|-----------|
| I0 | | |
| I1 | | N/A |
| I2 | | N/A |
| I3 | | |
| I4 | | N/A |
| I5 | | N/A |
| I6 | | |

5. 求指令所用的Cycle数（有forwarding）

6. 请调整顺序，尽可能减少stall（有forwarding），并求需要多少Cycle

汇编

(1) 用最少的指令实现 $x_{20} > x_{11} \mid x_{20} < 0$ 的跳转

(2) beq能跳转的范围比较小，请使用一对指令实现 `Beq x0, x1, L`（L较大）

(3)

```
int find(int *mv, int *v, int n) {
    if(n > 0) { *mv = max(v, n); return 1; }
    return -1;
}
int max(int *p, int n) {
    int i, m = p[0];
    for(int i = 1; i < n; i++) {
        if(p[i] > m) m = p[i];
    }
    return m;
}
```

将上述C语言代码转为RISCV汇编语言，要求进行保护（忘记题目的说法，总之要求在进入函数将要保护的寄存器push，离开前pop）。

存储层级相关

两部分，cache，TLB。

已知如下：

1. virtual memory address 54 bits, physical memory address 32 bits;
2. 4KB cache 2路组相连, block size 128B;
3. TLB 2路组相连, 512 entries;
4. page size 8KB。

题目

- (1) 求cache中Tag, Index, Block offset的位数, 求cache中一条data的位数。
(2) 求TLB中Tag, Index, Page offset的位数, 求一条TLB中存储data的位数
- 给出如下一组地址查询 (10进制), 请填入每一个地址hit/miss (使用LRU策略进行替换)

| 0 | 100 | 200 | 300 | 1024 | 2048 | 4096 | 250 | 100 |
|---|-----|-----|-----|------|------|------|-----|-----|
| | | | | | | | | |

根据上表求hit ratio; 求LRU替换多少个block

- 根据上边的查询, 给出最后cache中的内容 (仅需给出valid部分), 以如下格式: `<index, tag, data[xx bytes-xx bytes]>`

最后一次更改时间: 2022.06.20 13:23

在此感谢刘海风老师和郝家辉学长的帮助! 郝助教毕业快乐!

同时感谢热心的98ers的补充, 尤其感谢@[贝拉拉贝拉](#)提供的内容, 极大丰富了初版回忆卷。