需要做的事情：

1、补全函数

包括函数以及母函数中的接口

```
304     void BitTypeConvert_addi();
305     void BitTypeConvert_1();
306     void BitTypeConvert_2();
307     void BitTypeConvert_3();
308     void BitTypeConvert_4();
309     void BitTypeConvert_5();
310     void BitTypeConvert_6();
311     void BitTypeConvert_7();
312     void BitTypeConvert_8();
313     void BitTypeConvert_9();
314
315     void BitTypeConvert()
316     {
317         int opcode = getUValueFromBits(this->bi_instruction, 31, 26);
318
319         switch (opcode)
320         {
321         case 0b000000: // add
322             /* code */
323             break;
324
325         case 0b001000: // addi
326             BitTypeConvert_addi();
327             break;
328
329         default:
330             break;
331         }
332     }
```

```
334     void AsmTypeConvert_addi();
335     void AsmTypeConvert_1();
336     void AsmTypeConvert_2();
337     void AsmTypeConvert_3();
338     void AsmTypeConvert_4();
339     void AsmTypeConvert_5();
340     void AsmTypeConvert_6();
341     void AsmTypeConvert_7();
342     void AsmTypeConvert_8();
343     void AsmTypeConvert_9();
344
345     void AsmTypeConvert()
346     {
347         switch (map_of_asm[this->splt_instruction[0]])
348         {
349         case 0b000000: // add
350             /* code */
351             break;
352
353         case 0b001000: // addi
354             AsmTypeConvert_addi();
355             break;
356
357         default:
358             break;
359         }
360     }
361
```

```
395     void Exe_addi(RegAndMemory* target);
396     void Exe_1(RegAndMemory* target);
397     void Exe_2(RegAndMemory* target);
398     void Exe_3(RegAndMemory* target);
399     void Exe_4(RegAndMemory* target);
400     void Exe_5(RegAndMemory* target);
401     void Exe_6(RegAndMemory* target);
402     void Exe_7(RegAndMemory* target);
403     void Exe_8(RegAndMemory* target);
404     void Exe_9(RegAndMemory* target);
405
406     void Execute(RegAndMemory* target)
407     {
408         int opcode = Instruction::getUValueFromBits(this->bi_instruction, 31, 26
409
410         switch (opcode)
411         {
412         case 1:
413             /* code */
414             break;
415
416         case 0b001000: // addi
417             Exe_addi(target);
418             break;
419
420         default:
421             break;
422         }
423     }
424
425 };
```

具体补全补全在下面的实现

```
484  void Instruction::BitTypeConvert_1()
485  {}
486
487  void Instruction::AsmTypeConvert_1()
488  {}
489
490  void Instruction::Exe_1(RegAndMemory* target)
491  {}
492
493  void Instruction::BitTypeConvert_2()
494  {}
495
496  void Instruction::AsmTypeConvert_2()
497  {}
498
499  void Instruction::Exe_2(RegAndMemory* target)
500  {}
501
502  void Instruction::BitTypeConvert_3()
503  {}
504
505  void Instruction::AsmTypeConvert_3()
506  {}
507
508  void Instruction::Exe_3(RegAndMemory* target)
509  {}
510
511  void Instruction::BitTypeConvert_4()
512  {}
513
514  void Instruction::AsmTypeConvert_4()
515  {}
516
517  void Instruction::Exe_4(RegAndMemory* target)
518  {}
519
520  void Instruction::BitTypeConvert_5()
```

2、InitMap函数中加上对应指令的操作码

```
205      static void InitMap()
206      {
281          map_of_reg_str[31] = "$ra";
282
283          map_of_asm["add"] = 0b000000;
284          map_of_asm["addi"] = 0b001000;
285
286          map_of_opcode[0b000000] = "add";
287          map_of_opcode[0b001000] = "addi";
288      }
```

寄存器已经加好了，不用加了！

3、编写和调试请参考现有代码，getValueFromBits该函数还有bug，addi的对应操作没有考虑负数的情况，写的时候请务必注意！！！