COA2021-programming13

Good luck and have fun!

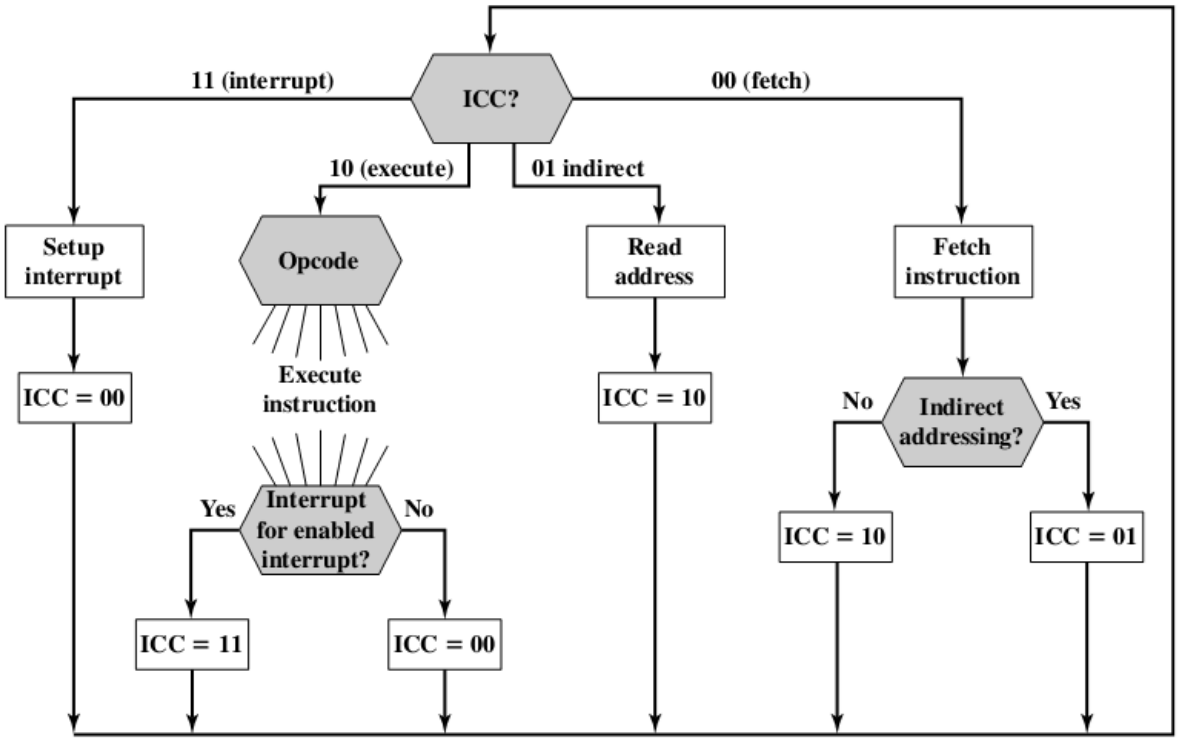
1 实验要求

实现指令控制状态机ICC：上一次作业只要求能够解析和执行单条指令，本次作业要求能够执行一连串的指令

为此需要实现：

1. 实现 *opcode=0xf4 hlt* 指令
2. 阅读 ./test 文件夹中的指令序列，实现其中的所有指令
3. 在已有代码基础上实现ICC状态机，允许对代码进行重构

在CPU中，指令的执行流一般是由ICC进行控制，如图所示，其中不需要考虑ICC=0b11的中断分支：



2 实验攻略

2.1 实验概述

作为本学期最后一次编程作业，本次作业主要考察的是ICC状态机的实现。ICC状态机会不断重复获取、解析、执行指令，直到执行到hlt指令为止(实际上hlt是关机指令，为了避免实现复杂的中断(ICC=0b11)，本作业简单地将hlt指定为指令终止指令)

在本次作业中，我们不限定具体的实现方式，你可以使用一个最简单的while循环，也可以考虑状态设计模式，或者一个专门的控制器类。

2.2 代码导读

2.2.1 代码结构

```
.
|  .gitignore
|  pom.xml
|  README.md
|
├─src
|  ├─main
|  |  └─java
|  |      ├─cpu
|  |      |  |  CPU.java       # CPU类，需要阅读
|  |      |  |  CPU_State.java # CPU的寄存器列表
|  |      |  |  MMU.java
|  |      |  |
|  |      |  ├─alu
|  |      |  |
|  |      |  ├─instr
|  |      |  |  ├─all_instrs # 需要在这个文件夹里面添加指令的实现类
|  |      |  |  |      Add.java          # ADD指令的实现
|  |      |  |  |      InstrFactory.java # 指令工厂，使用了Java反射机制
|  |      |  |  |      Instruction.java  # 指令接口，需要阅读
|  |      |  |  |      Opcode.java       # 指令的操作码，参考Intel 32位指令实现
|  |      |  |  |
|  |      |  |  └─decode # 取指令相关的类，需要阅读
|  |      |  |
|  |      |  ├─registers # 寄存器类的定义
|  |      |  |
|  |      |  └─utils
|  |      |
|  |      ├─kernel # 测试时的程序入口
|  |      |      Loader.java
|  |      |      MainEntry.java
|  |      |
|  |      ├─memory
|  |      |
|  |      ├─program
|  |      |      Log.java # 实现控制台输出
|  |      |
|  |      ├─transformer
|  |      |
|  |      └─util
|  |
|  └─test
|      └─java
|          └─cpu
|              └─instr
|                      ICCTest.java # 测试类
|
└─test # 6个测试用例分别使用到的指令序列，需要自行阅读
        icc_test_1.txt
        icc_test_2.txt
        icc_test_3.txt
        icc_test_4.txt
        icc_test_5.txt
```

```
54          icc_test_6.txt
55
```

2.3 实现指导

2.3.1 指令描述

完整的指令描述需要通过查阅i386手册获得。

2.3.2 实现参考

为了方便大家实现具体的指令类，我们已经实现好了一个ADD指令供大家进行参考。已经实现好的指令描述如下：

- opcode=0x05 ADD EAX, imm32
    - 指令结构：1字节opcode + 4字节立即数imm
    - 功能：DEST = SRC + imm32;
    - 目的操作数DEST：EAX寄存器中的值
    - 源操作数SRC：EAX寄存器中的值

3 参考资料

英特尔80386程序员参考手册(i386)intel：https://css.csail.mit.edu/6.858/2014/readings/i386.pdf

One-Byte Opcode Map

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ADD						PUSH	POP	OR						PUSH	2-byte
	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	ES	ES	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	CS	escape
1	ADC						PUSH	POP	SBB						PUSH	POP
	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	SS	SS	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	DS	DS
2	AND						SEG	DAA	SUB						SEG	DAS
	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	=ES		Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	=CS	
3	XOR						SEG	AAA	CMP						SEG	AAS
	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	=SS		Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	AL,Ib	eAX,Iv	=CS	
4	INC general register								DEC general register							
	eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI	eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI
5	PUSH general register								POP into general register							
	eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI	eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI
6	PUSHA	POPA	BOUND Gv,Ma	ARPL Ew,Rw	SEG =FS	SEG =GS	Operand Size	Address Size	PUSH Ib	IMUL GvEvIv	PUSH Ib	IMUL GvEvIv	INSB Yb,DX	INSW/D Yb,DX	OUTSB Dx,Xb	OUTSW/D DX,Xv
7	Short displacement jump of condition (Jb)								Short-displacement jump on condition(Jb)							
	JO	JNO	JB	JNB	JZ	JNZ	JBE	JNBE	JS	JNS	JP	JNP	JL	JNL	JLE	JNLE
8	Immediate Grp1			Grp1	TEST		XCNG		MOV				MOV	LEA	MOV	POP
	Eb,Ib	Ev,Iv		Ev,Iv	Eb,Gb	Ev,Gv	Eb,Gb	Ev,Gv	Eb,Gb	Ev,Gv	Gb,Eb	Gv,Ev	Ew,Sw	Gv,M	Sw,Ew	Ev
9	NOP	XCHG word or double-word register with eAX							CBW	CWD	CALL Ap	WAIT	PUSHF Fv	POPF Fv	SAHF	LAHF
		eCX	eDX	eBX	eSP	eBP	eSI	eDI								
A	MOV				MOVSB	MOVSW/D	CMPSB	CMPSW/D	TEST		STOSB	STOSW/D	LODSB	LODSW/D	SCASB	SCASW/D
	AL,Ob	eAX,Ov	Ob,AL	Ov,eAX	Xb,Yb	Xv,Yv	Xb,Yb	Xv,Yv	AL,Ib	eAX,Iv	Yb,AL	Yv,eAX	AL,Xb	eAX,Xv	AL,Xb	eAX,Xv
B	MOV immediate byte into byte register								MOV immediate word or double into word or double register							
	AL	CL	DL	BL	AH	CH	DH	BH	eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI
C	Shift Grp2		RET near		LES	LDS	MOV		ENTER	LEAVE	RET far		INT	INT	INTO	IRET
	Eb,Ib	Ev,Iv	Iw		Gv,Mp	Gv,Mp	Eb,Ib	Ev,Iv	Iw,Ib		Iw		3	Ib		
D	Shift Grp2				AAM	AAD		XLAT	ESC(Escape to coprocessor instruction set)							
	Eb,1	Ev,1	Eb,CL	Ev,CL												
E	LOOPNE	LOOPE	LOOP	JCXZ	IN		OUT		CALL	JNP			IN		OUT	
	Jb	Jb	Jb	Jb	AL,Ib	eAX,Ib	Ib,AL	Ib,eAX	Av	Jv	Ap	Jb	AL,DX	eAX,DX	DX,AL	DX,eAX
F	LOCK		REPNE	REP REPE	HLT	CMC	Unary Grp3		CLC	STC	CLI	STI	CLD	STD	INC/DEC Grp4	Indirct Grp5
							Eb	Ev								