

实验二：DLX/MIPS/RISC-V 指令格式

19281171 王雨潇

实验报告要求：实验报告用 Word 排版，文件内一定要有"学号姓名_学号"。实验报告应提交到课程平台，按时提交。实验报告的内容应尽量全面，避免简单化。简述实验内容，实验过程，实验结果，实验分析，心得体会等。答每道题前应将题目拷贝到实验报告上。

一、 实验内容

本次实验的主要目的是熟悉 DLX-MIPS 的指令格式，并初步了解 RISC-V 的指令格式。

二、 实验过程

实验分为两个部分：

1. 熟悉 winDLX 模拟器，并完成以下实验及分析：

A. 参考提供的资料，学习 winDLX 模拟器的各窗口及各项功能。

B. 运行 FACT.S, GCM.S, PRIM.S 程序，了解各程序的功能。要求详细分析 PRIM.S 程序的指令及功能（Obersve the memory areas in bytes for prime numbers）。

C. 对于 FACT.S 程序，请从中选出 3-5 条不同的指令，并分别指出它是哪种指令（R-type, I-type, 还是 J-type），并参照教科书 2-28 页中的图 2.19（如下所示），填写指令格式中各个域的二进制值。为了清楚起见，最好用填表的形式。注：分析指令各域应该填写数值的原因。

Format	Bits					
	0 5	6 10	11 15	16 20	21 25	26 31
R-type	0x0	Rs1	Rs2	Rd	<i>unused</i>	<i>opcode</i>
I-type	<i>opcode</i>	Rs1	Rd	<i>immediate</i>		
J-type	<i>opcode</i>	<i>value</i>				

D. 请通过 CODE 窗口的各种选项查看数据，进而判断出 DLX 是 big-endian 还是 little-endian，为什么（结合实验中的观察分析）？是否是对齐的（aligned）？

2. 了解 RISC-V 的在线模拟器 Venus 的使用方法。包括如下内容：

A. 参考提供的资料，学习 RISC-V 的在线模拟器 Venus 的用法。

B. 程序 ex1.s 的功能是什么？程序运行完后，得出的结果是什么？

C. 对于 ex1.s 程序，请从中选出 3-5 条不同的指令，并对于其中每条指令，指出它是哪种指令。请注意：RISC-V 的指令格式分 6 种（除了 R-type, I-type, J-type，还有 U-type, S-type, B-type），并填写指令格式中各个域的二进制值。

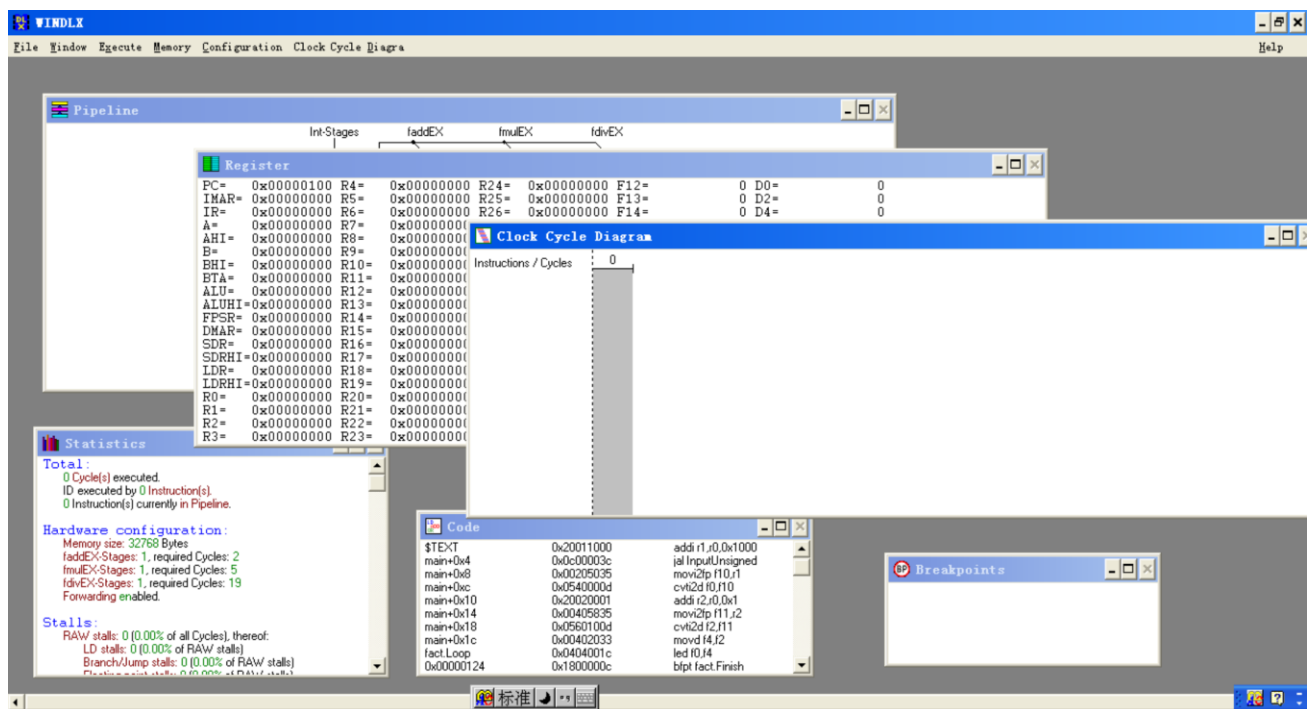
31	30	25	24	21	20	19	15	14	12	11	8	7	6	0	
funct7				rs2		rs1	funct3		rd			opcode		R-type	
imm[11:0]						rs1	funct3		rd			opcode		I-type	
imm[11:5]				rs2		rs1	funct3		imm[4:0]			opcode		S-type	
imm[12]		imm[10:5]		rs2		rs1	funct3		imm[4:1]		imm[11]		opcode		B-type
imm[31:12]									rd			opcode		U-type	
imm[20]		imm[10:1]			imm[11]		imm[19:12]			rd			opcode		J-type

三、实验结果与分析

1. winDLX 模拟器部分

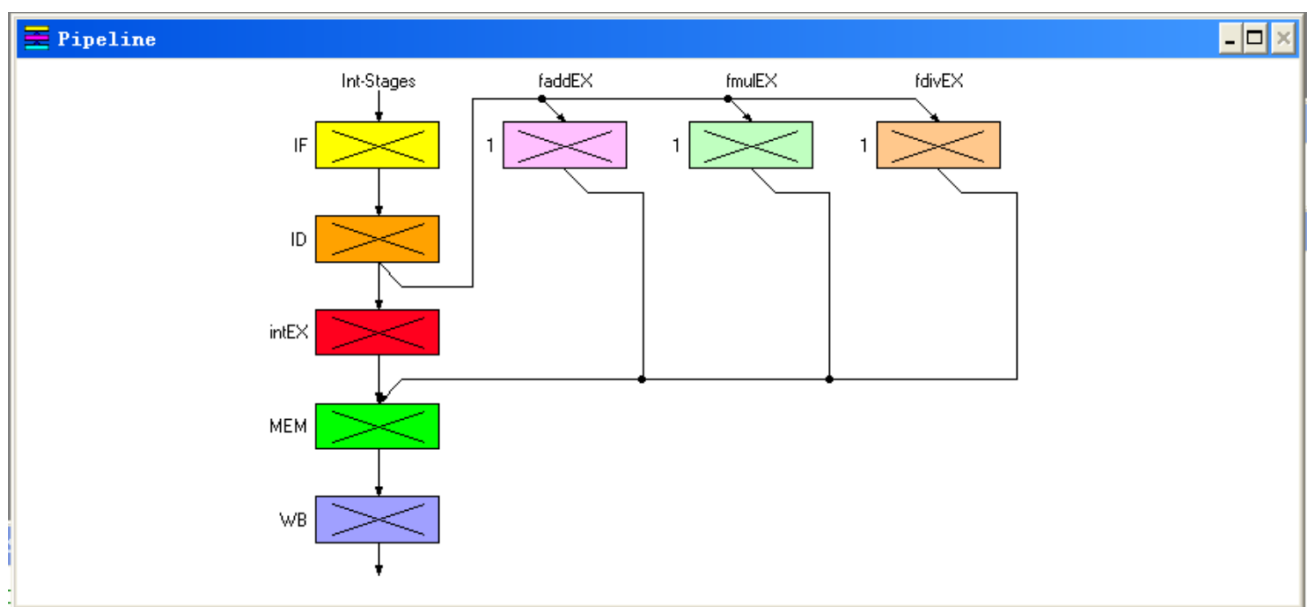
A. 参考提供的资料，学习 winDLX 模拟器的各窗口及各项功能。

答：如下图所示，winDLX 模拟器的初始窗口共有 6 个：



其中:

(1) Pipeline 窗口显示 DLX 的流水段和操作单元情况;



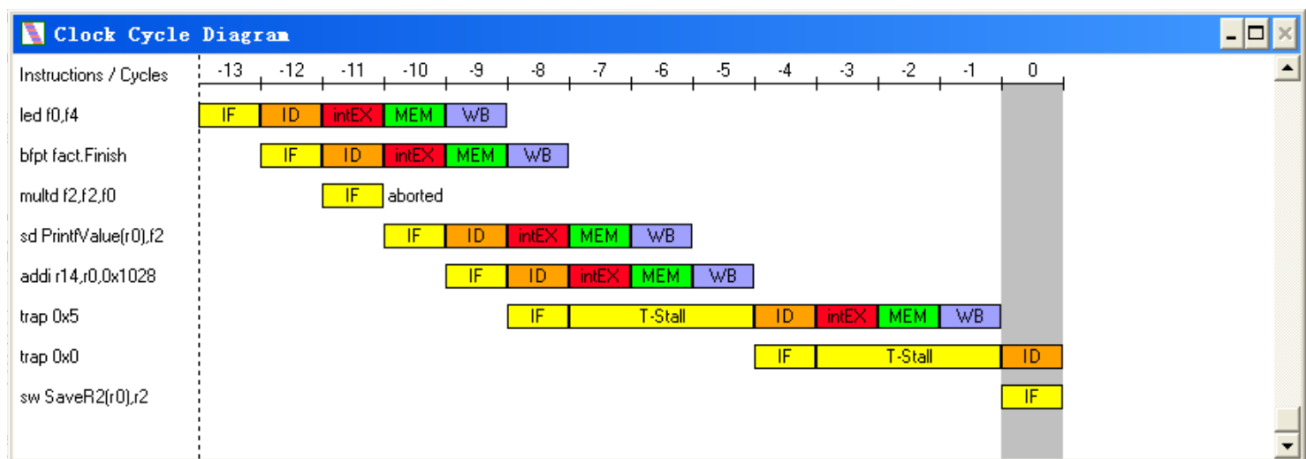
(2) Register 窗口显示各寄存器内容;

Register									
PC=	0x00000100	R8=	0x00000000	F0=	0	F24=	0		
IMAR=	0x00000000	R9=	0x00000000	F1=	0	F25=	0		
IR=	0x00000000	R10=	0x00000000	F2=	0	F26=	0		
A=	0x00000000	R11=	0x00000000	F3=	0	F27=	0		
AHI=	0x00000000	R12=	0x00000000	F4=	0	F28=	0		
B=	0x00000000	R13=	0x00000000	F5=	0	F29=	0		
BHI=	0x00000000	R14=	0x00000000	F6=	0	F30=	0		
BTA=	0x00000000	R15=	0x00000000	F7=	0	F31=	0		
ALU=	0x00000000	R16=	0x00000000	F8=	0	D0=	0		
ALUHI=	0x00000000	R17=	0x00000000	F9=	0	D2=	0		
FPSR=	0x00000000	R18=	0x00000000	F10=	0	D4=	0		
DMAR=	0x00000000	R19=	0x00000000	F11=	0	D6=	0		
SDR=	0x00000000	R20=	0x00000000	F12=	0	D8=	0		
SDRHI=	0x00000000	R21=	0x00000000	F13=	0	D10=	0		
LDR=	0x00000000	R22=	0x00000000	F14=	0	D12=	0		
LDRHI=	0x00000000	R23=	0x00000000	F15=	0	D14=	0		
R0=	0x00000000	R24=	0x00000000	F16=	0	D16=	0		
R1=	0x00000000	R25=	0x00000000	F17=	0	D18=	0		
R2=	0x00000000	R26=	0x00000000	F18=	0	D20=	0		
R3=	0x00000000	R27=	0x00000000	F19=	0	D22=	0		
R4=	0x00000000	R28=	0x00000000	F20=	0	D24=	0		
R5=	0x00000000	R29=	0x00000000	F21=	0	D26=	0		
R6=	0x00000000	R30=	0x00000000	F22=	0	D28=	0		
R7=	0x00000000	R31=	0x00000000	F23=	0	D30=	0		

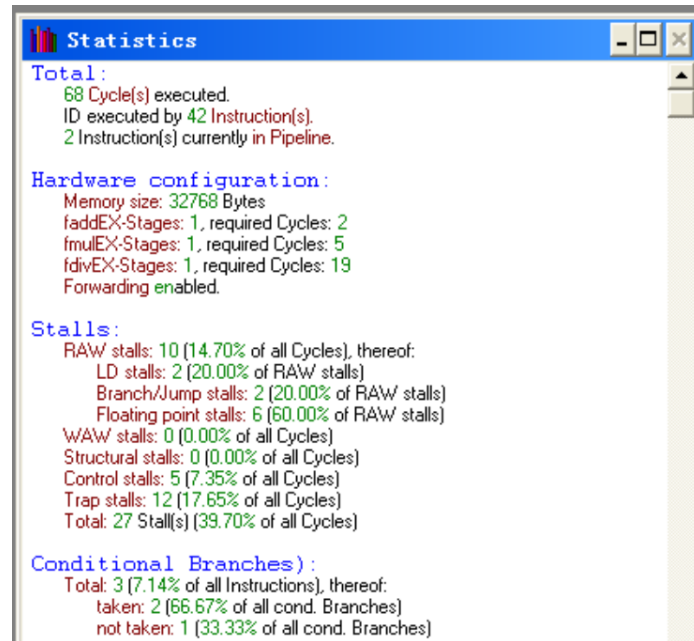
(3) Code 窗口每行显示地址，机器码，汇编代码，可以按 F7 单步调试

Code		
0x00000128	0x04401006	multd f2,f2,f0
0x0000012c	0x04040005	subd f0,f0,f4
0x00000130	0x0bffffec	j fact.Loop
0x00000134	0xbc02102c	sd PrintfValue(r0),f2
0x00000138	0x200e1028	addi r14,r0,0x1028
0x0000013c	0x44000005	trap 0x5
0x00000140	0x44000000	ID trap 0x0
0x00000144	0xac021094	IF sw SaveR2(r0),r2
0x00000148	0xac031098	sw SaveR3(r0),r3
0x0000014c	0xac04109c	sw SaveR4(r0),r4

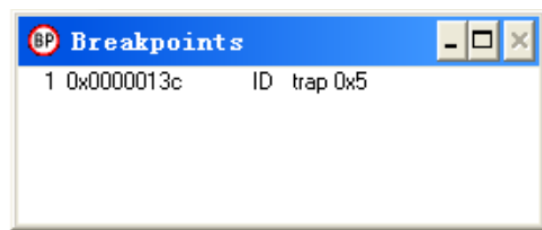
(4) Clock Cycle Diagram 窗口显示时空图



(5) Statistics 窗口是统计信息

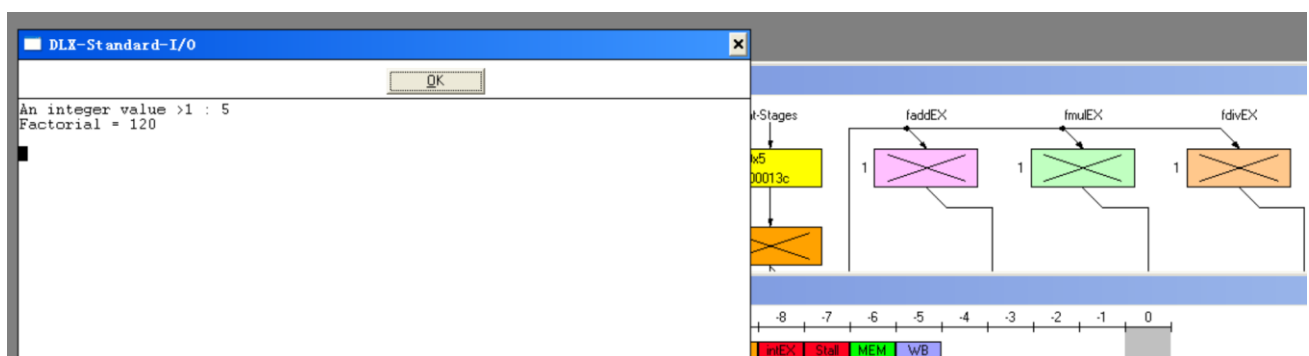


(6) Breakpoints 用于显示设置的断点

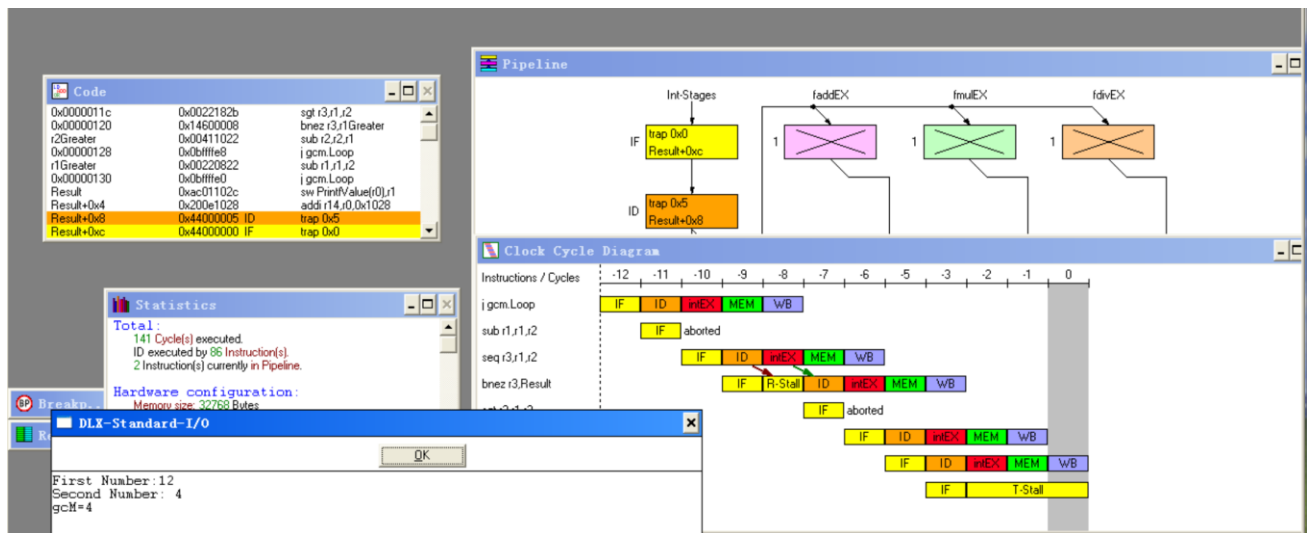


B. 运行 FACT.S, GCM.S, PRIM.S 程序，了解各程序的功能。要求详细分析 PRIM.S 程序的指令及功能 (Obersve the memory areas in bytes for prime numbers)。

答：FACT.S 的功能是计算一个整数的阶乘；



GCM.S 的功能是求两个整数的最大公约数；



PRIM.S 程序的功能是求素数，详细分析如下

(1) 程序源码、以及我添加的注释如下所示：

```

;***** WINDLX Exp.2: Generate prime number table *****
;***** (c) 1991 Günther Raidl *****
;***** Modified 1992 Maziar Khosravipour *****

;-----
; Program begins at symbol main
; generates a table with the first 'Count' prime numbers from 'Table'
;-----

.data
;*** size of table
.global Count
Count:
.word 10
.global Table
Table:
.space Count*4
; wyx: 此处定义两个全局变量 Count 和 Table, 初始值分别为 10 和 40

.text
.global main
main:
;*** Initialization
addi r1,r0,0 ;Index in Table
addi r2,r0,2 ;Current value

```

```
; wyx: 进入主程序, 首先令 R1 = 0, R2 = 2
; 下方自带注释说明目标是判断 R2 能不能被 Table 中的数值整除
```

```
*** Determine, if R2 can be divided by a value in table
```

```
NextValue:
```

```
addi    r3,r0,0    ;Helpindex in Table
```

```
Loop:
```

```
seq     r4,r1,r3    ;End of Table?
bnez    r4,IsPrim   ;R2 is a prime number
lw      r5,Table(R3)
divu    r6,r2,r5
multu   r7,r6,r5
subu    r8,r2,r7
beqz    r8,IsNoPrim
addi    r3,r3,4
j       Loop
```

```
; wyx: R3 是数组 Table 的下标, 此处循环 Table(R3) 就是从 0 开始遍历整个数组
; 每次循环: loop {R5 = Table[R3], R6 = R2/R5, R7 = R6*R5, R8 = R2-R7}
; 可以看出 R8 是求 R2 % Table[R3] 的余数,
; 若 R8 == 0, 则 R2 被整除 (可下结论 R2 不是质数);
; 反之, 则 R2 没被整除;
; 程序如果能运行到 R1 == R3, 说明整个数组遍历了都不能整除 R2, 即 R2 是质数
```

```
IsPrim:    *** Write value into Table and increment index
```

```
sw      Table(r1),r2
addi    r1,r1,4
```

```
*** 'Count' reached?
```

```
lw      r9,Count
srli    r10,r1,2
sge     r11,r10,r9
bnez    r11,Finish
```

```
; wyx: 找到质数就记录在 Table[R1] 中, 找够 Count 个即停止
```

```
IsNoPrim:  *** Check next value
```

```
addi    r2,r2,1    ;increment R2
j       NextValue
```

```
; wyx: 没找到就 R2 += 1 继续找下一个
```

```
Finish:    *** end
```

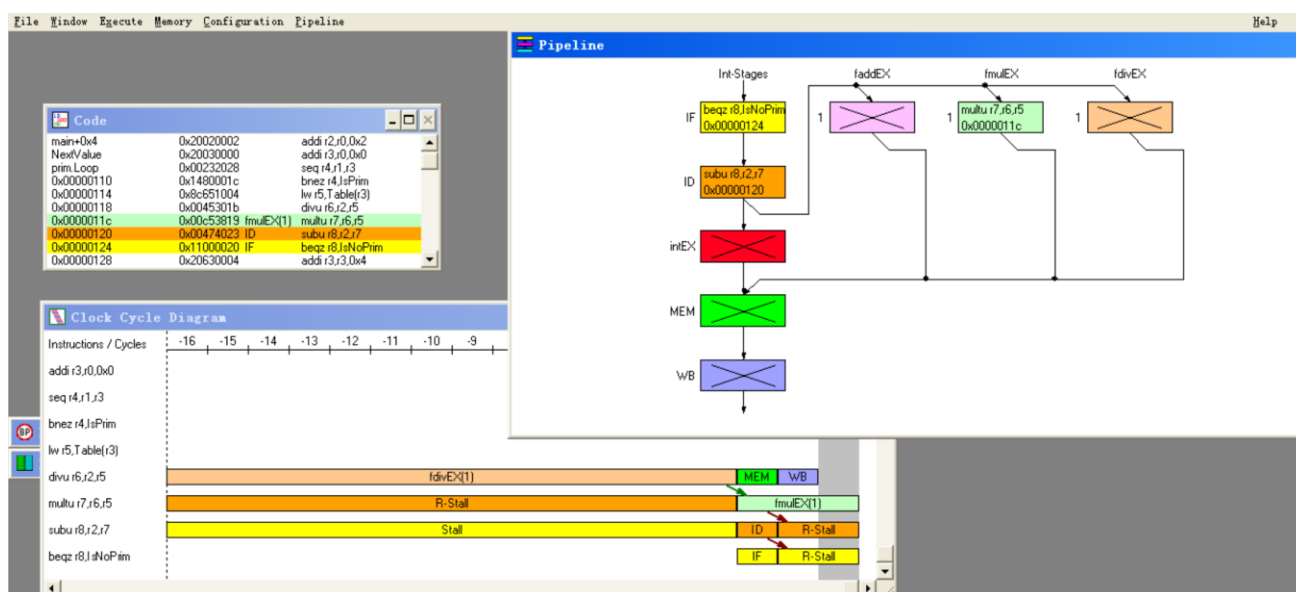
```
trap    0
```

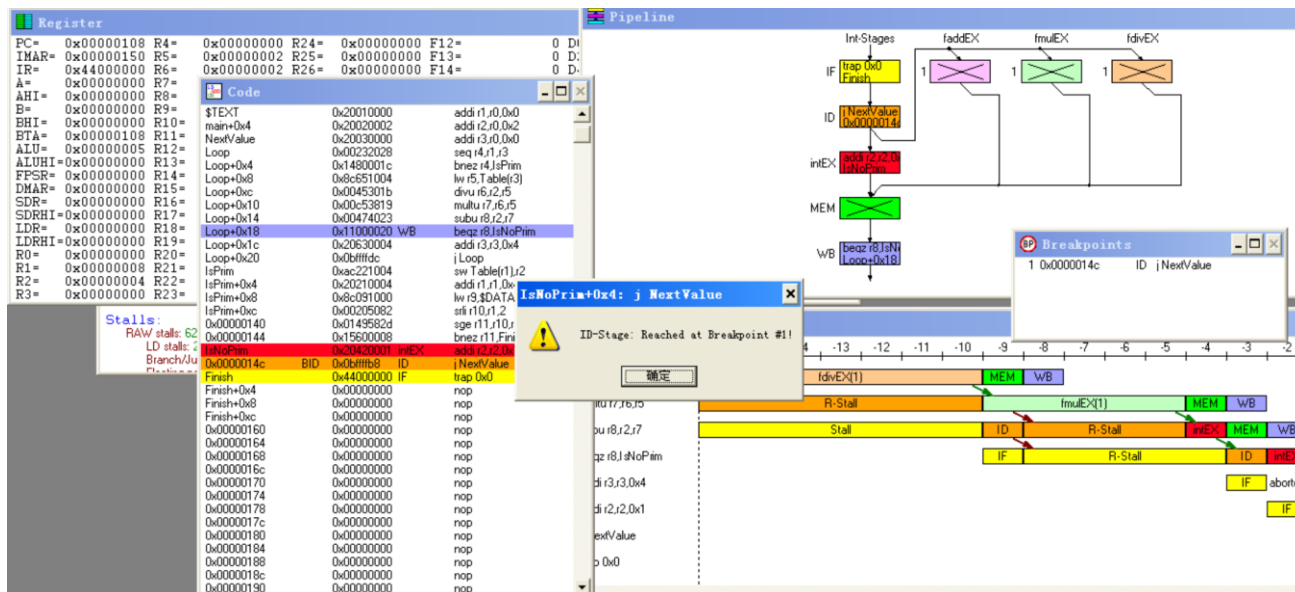
(2) 程序运行过程

从各 R 寄存器变化规律，可以验证之前阅读源码的结果；

Register											
PC=	0x00000124	R4=	0x00000000	R24=	0x00000000	F12=					
IMAR=	0x00000120	R5=	0x0000000b	R25=	0x00000000	F13=					
IR=	0x00474023	R6=	0x00000002	R26=	0x00000000	F14=					
A=	0x00000002	R7=	0x0000000e	R27=	0x00000000	F15=					
AHI=	0x00000000	R8=	0x00000005	R28=	0x00000000	F16=					
B=	0x0000000b	R9=	0x0000000a	R29=	0x00000000	F17=					
BHI=	0x00000000	R10=	0x00000007	R30=	0x00000000	F18=					
BTA=	0x00000000	R11=	0x00000000	R31=	0x00000000	F19=					
ALU=	0x00000000	R12=	0x00000000	F0=		0 F20=					
ALUHI=	0x00000000	R13=	0x00000000	F1=		0 F21=					
FPSR=	0x00000000	R14=	0x00000000	F2=		0 F22=					
DMAR=	0x00001014	R15=	0x00000000	F3=		0 F23=					
SDR=	0x00000000	R16=	0x00000000	F4=		0 F24=					
SDRHI=	0x00000000	R17=	0x00000000	F5=		0 F25=					
LDR=	0x0000000b	R18=	0x00000000	F6=		0 F26=					
LDRHI=	0x00000000	R19=	0x00000000	F7=		0 F27=					
R0=	0x00000000	R20=	0x00000000	F8=		0 F28=					
R1=	0x0000001c	R21=	0x00000000	F9=		0 F29=					
R2=	0x00000013	R22=	0x00000000	F10=		0 F30=					
R3=	0x00000010	R23=	0x00000000	F11=		0 F31=					

单步执行过程中发现，每个循环里最耗时的操作是计算除法和乘法，周期非常长；





C. 对于 FACT.S 程序, 请从中选出 3-5 条不同的指令, 并分别指出它是哪种指令 (R-type, I-type, 还是 J-type), 并参照教科书 2-28 页中的图 2.19 (如下所示), 填写指令格式中各个域的二进制值。为了清楚起见, 最好用填表的形式。

答: 指令各域应该填写数值的原因: 数值机器码才是计算机能理解的信息;

指令截图	比特位				结论
lw r2,SaveR2[r0] Adr.: input.Finish Code: 0x8c021060	1000 11 (opcode)	00 000 (Rs1)	0 0010 (Rd)	0001 0000 0110 0000 (imm)	应为 I-type 指令
addi r1,r0,0x1070 Adr.: main Code: 0x20011070	0010 00 (opcode)	00 000 (Rs1)	0 0001 (Rd)	0001 0000 0111 0000 (imm)	应为 I-type 指令
jr r31 Adr.: input.Finish+0x10 Code: 0x4be00000	0100 10 (opcode)	11 1110 0000 0000 0000 0000 0000 (value)			应为 J-type 指令

D. 请通过 CODE 窗口的各种选项查看数据, 进而判断出 DLX 是 big-endian 还是 little-endian, 为什么 (结合实验中的观察分析)? 是否是对齐的 (aligned)?

答: 用两个程序段来验证这个问题:

(1) 首先, 以 GCM.S 程序为例, 发现 Prompt1 和 Prompt2 变量在内存中的存储方式是

```

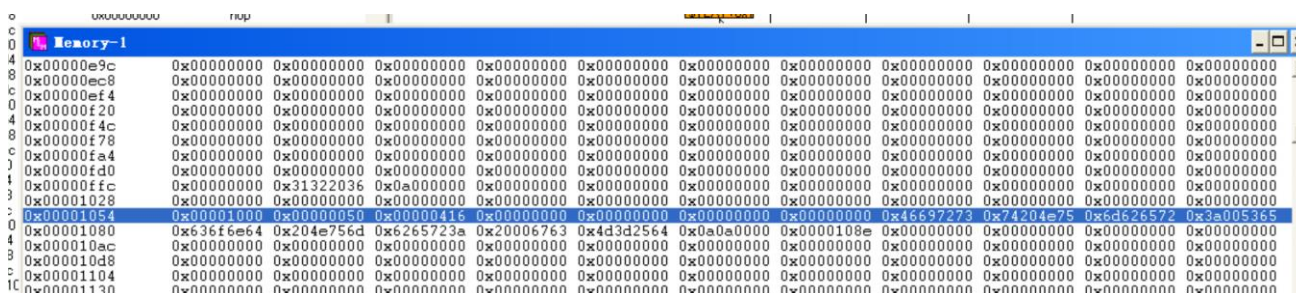
13
14 ;*** Prompts for input
15 Prompt1: .asciiz "First Number:"
16 Prompt2: .asciiz "Second Number: "
17

```

```

0x00000000 0x00000000 0x00000000 0x00000000
0x46697273 0x74204e75 0x6d626572 0x3a005365
0x00000000 0x00000000 0x00000000 0x00000000

```



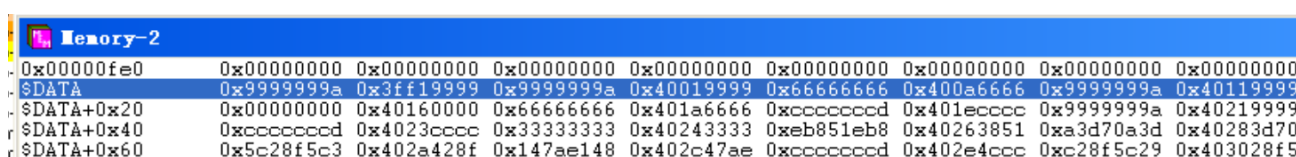
对比 ascii 码参考表：

对应字符	F	i	r	s	t	空格	N	u	:	NUL	S	e
十六进制表示	46	69	72	73	74	20	4e	75	3a	00	53	65

(2) 再编写一句数据段定义代码：

```
.double 1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.10,11.11,12.12,13.13,14.14,15.15,16.16
```

在内存中表现为下图所示



综合以上两例，发现 DLX 架构是大端序的、对齐的。

2. Venus 模拟器部分

了解 RISC-V 的在线模拟器 Venus 的使用方法。包括如下内容：

A. 参考提供的资料，学习 RISC-V 的在线模拟器 Venus 的用法。

VenusEditorSimulatorChocopy

TerminalFilesURLWikiJVM

Venus Web Terminal

Fri Apr 08 2022 17:22:17 GMT+0800 (中国标准时间)

Enter "help" for more information.

[user@venus] /# help

assemblecdclockdateedithelp

catclearcpdownloadexit

[user@venus] /#

Settings

GeneralCalling ConventionTracerPackages

Simulator Default Args

Text Start0x00000000

Max History-1

Save on CloseSave on Close

Aligned AddressingForce Aligned Addressing?

Mutable TextMutable Text?

Only Ecall ExitOnly Ecall Exit?

Default Reg StatesSet Registers on Init?

Allow AccessAllow Access Between Stack and Heap?

Max number of steps: (Negative means ignored)-1

VenusEditorSimulatorChocopy

RunStepPrevResetDumpTrace

PC	Machine Code	Basic Code	Original Code
0x0	0x000002B3	add x5 x0 x0	main: add t0, x0, x0
0x4	0x00100313	addi x6 x0 1	addi t1, x0, 1
0x8	0x10000E17	auipc x28 65536	la t3, n
0xc	0x008E0E13	addi x28 x28 8	la t3, n
0x10	0x000E2E03	lw x28 0(x28)	lw t3, 0(t3)
0x14	0x000E0C63	beq x28 x0 24	fib: beq t3, x0, finish
0x18	0x005303B3	add x7 x6 x5	add t2, t1, t0
0x1c	0x00030293	addi x5 x6 0	mv t0, t1
0x20	0x00000000	add x6 x7 0	mv t1, t2

Copy!Download!Clear!

console output

(x4)

t0(x5)0x00000000

t1(x6)0x00000001

t2(x7)0x00000000

s0(x8)0x00000000

s1(x9)0x00000000

a0(x10)0x00000000

DisplayHex

B. 程序 ex1.s 的功能是什么？程序运行完后，得出的结果是什么？

答： 源程序得出的结果是斐波那契数列（下标从 0 开始的）第 9 个数， 34

0x1c	0x00030293	addi x5 x6 0	mv t0, t1	s0 (x8)
0x20	0x00030212	addi x6 x7 0	mv t1, t0	s1
Copy! Download! Clear!				Display Settings
34				

对代码段的分析注释如下：

```
.data
.word 2, 4, 6, 8
n: .word 9

.text
main:  add    t0, x0, x0
      addi   t1, x0, 1
      la     t3, n
      lw     t3, 0(t3)
fib:   beq    t3, x0, finish # 循环下列过程，运行到下标 n 为止
      add    t2, t1, t0      # t2 = t1 + t0
      mv     t0, t1          # t0 = t1
      mv     t1, t2          # t1 = t2
      addi   t3, t3, -1      # t3--
      j      fib
finish: addi   a0, x0, 1      # a0 = 1, a1 = t0
      addi   a1, t0, 0
      ecall  # print integer ecall
      addi   a0, x0, 10
      ecall  # terminate ecall
```

再通过分别更改变量 n 的值为 10, 11, 12，验证程序的功能为计算斐波那契数列，并在控制台输出数列下标为 n 的数字；

C. 对于 ex1.s 程序，请从中选出 3-5 条不同的指令，并对于其中每条指令，指出它是哪种指令。

指令	比特位						结论
addi x6 x0 1	0000 0000 0001(imm)	0000 0(Rs1)	000(add)	0011 0(Rd)	001 0011(opcode)		I- type
auipc x28 65536	0001 0000 0000 0000 0000(imm)			1110 0(Rd)	001 0111(opcode)		U- type
lw x28 0(x28)	0000 0000 0000(imm)		1110 0(Rs1)		010(lw)	1110 0(Rd) 000 0011(opcode)	I- type

四、 心得体会

本次实验我通过 WinDLX 和 Venus 模拟器，初步了解了 DLX 和 RISC-V 指令的格式。通过借助在模拟器上的实际操作过程，观察代码的单步执行结果，我们可以更深入的理解指令集体系结构，掌握识别指令类型的技能。