0.1 2025 年电子信息与通信学院计算机组成原理与接口技术考试卷回忆版初稿

1 填空题

1. 单精度浮点数浮点数 3.5 存储需要 _____ 字节，十六进制表示为 _____。其存储于 0x10，若计算机是小端序，0x12 是 _____；若计算机是大端序，0x10 是 _____。

2. C 语言程序溢出

```
char a = -56, b = -78;
char sum = a + b, diff = a - b;
```

   写出 a, b, sum, diff 的十六进制表示。

3. 计算机组成结构计算机的基本组成结构为 （五个空）

4. 特殊寄存器 PC 计算机中存储当前程序位置（?）的特殊寄存器是 _____

5. 单精度浮点数已知十六进制 0x409c0000 ， 求浮点数 _____
   答案：*4.875*

2 MIPS 汇编程序

2.1 给出一段汇编程序

```
.data
int: .word 2025
.text
main:
la $s0, int
lw $a0, 0($s0)
jal sub
add $a0, $0, $v0
addi $v0, $0, 1
syscall
addi $v0, $0, 10
syscall

sub:
addi $sp, $sp, -12
sw $a0, 8($sp)
sw $ra, 4($sp)
sw $s0, 0($sp)
addi $t0, $0, 10  # 这里是这样的吗?
sub $t1, $a0, $t0
blez $t1, label1  # 这里似乎有 bug，应当为 bltz
div $a0, $t0
mflo $a0
mfhi $s0
jal sub
```

```
add $v0, $v0, $s0
j rel
label1:
add $v0, $0, $a0
rel:
lw $a0, 8($sp)
lw $ra, 4($sp)
lw $s0, 0($sp)
addi $sp, $sp, 12
jr $ra
```

1. 求汇编程序输出 （答案似乎为 *9*，十进制数位相加）
2. 求调用过程中栈的十六进制数（3x4）

基址	+8	+4	+0
0xFFFFFFF0			
0xFFFFFFE4			
0xFFFFFFD8			
0xFFFFFFCC			

2.2 给出一段 C 程序，将 str 的大小写翻转。

处理器为支持 add, addi, lw, sw, beq, j 等指令的 MIPS CPU。

```c
void sub(char *p){
    while(*p){
        if(*p >= 65 && *p <= 90)
            *p += 32;
        else if(*p >= 97 && *p <= 122)
            *p -= 32;

        p++;
    }
}

int main() {
    char str[] = "Hello, World!";
    sub(str);
    printf("%s\n", str);
    return 0;
}
```

```
.data
str: .asciiz "Hello, World!"
.text
main:
la $a0, str
jal sub
addi $v0, $0, 4
```

```
syscall
addi $v0, $0, 10
syscall

sub:
addi $sp, $sp, ___
sw $a0, 8($sp)
sw $ra, 4($sp)
sw $sp, 0($sp)
loop:
____ $t0, 0($a0)
beq $t0, $0, rel
addi $t1, $t0, -65
____ $t1, else
addi $t1, $t0, -90
____ $t1, else
addi $t0, $t0, 32
j next
else:
addi $t1, $t0, -97
____ $t1, next
addi $t1, $t0, -122
____ $t1, next
addi $t0, $t0, -32
next:
____ $t0, 0($a0)
addi $a0, $a0, 1
j loop
rel:
lw $a0, 8($sp)
lw $ra, 4($sp)
lw $sp, 0($sp)
addi $sp, $sp, ___
____ $ra
```

MIPS 汇编挖空，立即数使用十进制。
空为两个 `addi $sp, $sp` 的立即数，几个 `bltz bgtz`，`lb sb jr`

3 MIPS CPU

书上单周期 MIPS CPU 框图。

1. Instr 连接，写出 A-G 对应的地址 `Instr[x:y]`

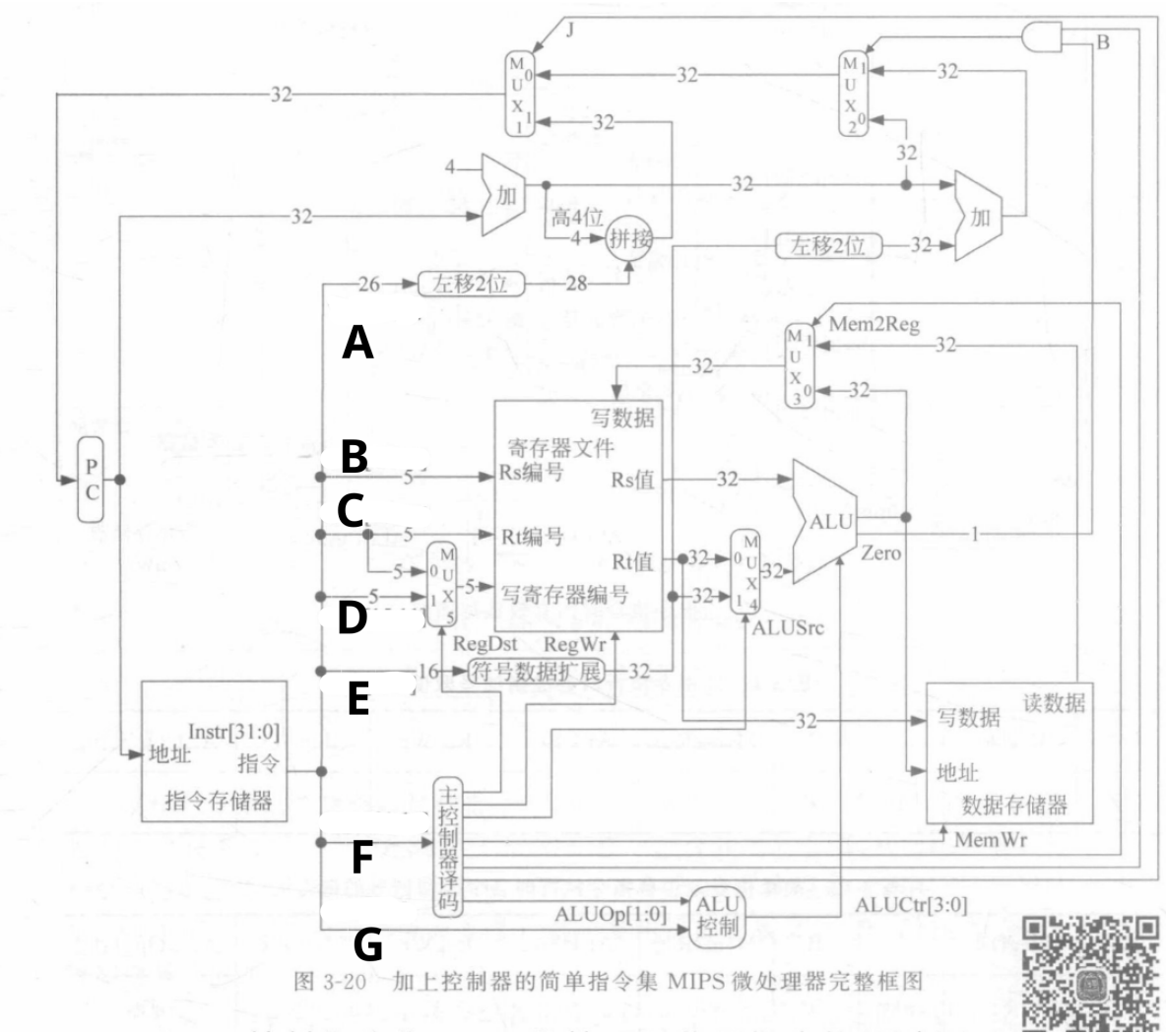

图 3-20 加上控制器的简单指令集 MIPS 微处理器完整框图

Figure 1: CPU

- A = `Instr[25:0]`
- B = `Instr[25:21]`
- C = `Instr[20:16]`
- D = `Instr[15:10]`
- E = `Instr[15:0]`
- F = `Instr[31:26]`
- G = `Instr[5:0]`

2. 波形图反推 MIPS 指令，写出十六进制**机器码**与**指令**。

   波形图除了 PC 为十进制，其他数据均为十六进制。半高线为信号 X。

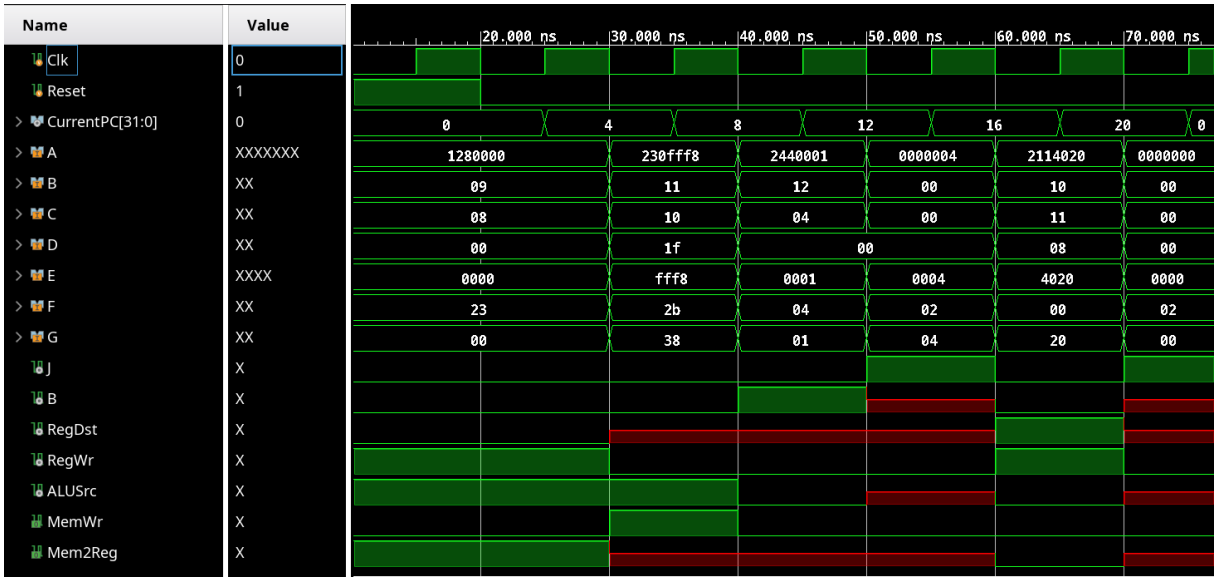

Figure 2: 波形图

```
# 我的答案， 问号 为不记得的
L0: lw $8, 0($9)
L1: sw $16, -8($?)
L2: beq $?, $?, L4
L3: j L4
L4: add $?, $?, $?
L5: j L0
```

4 存储系统

八个 62256 芯片（2M x 8bits）构成的多类型数据访问存储器接口，支持 32,16,8 位数据访问，地址为 0x0C000000~0x0CFFFFFF

1. 信号填空（四个数据线、一个地址线、一个片选线、一个写使能信号、N 个地址片选）
   D[31:24] D[23:16] D[15:8] D[7:0]（注意顺序）BE0 A[22:2] A23 A28~24 注意 A26 A27
2. 组成的存储器大小 _____ x _____ bits
3. 2G 内存，256K 缓存，行大小 128B。若采用 8 路组相连映射，求内存地址结构、缓存行构成（名称、位宽）。写出运算过程。
4. 同（3），若采用直接映射，求内存地址结构、缓存行构成（名称、位宽），以及缓存行数据使用率（百分比，两位小数）。写出运算过程。

5　并行 IO 接口中断程序

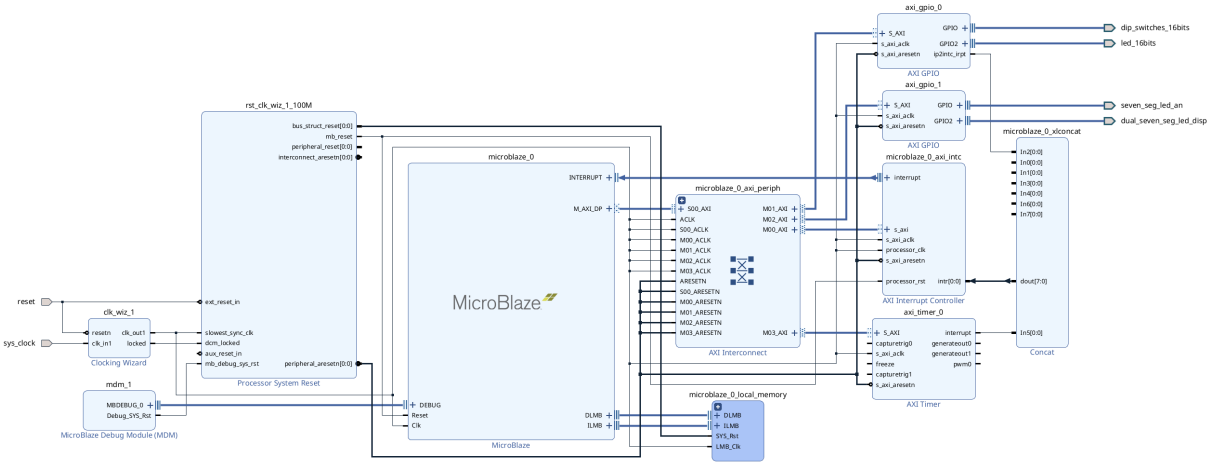

Figure 3: 硬件框图

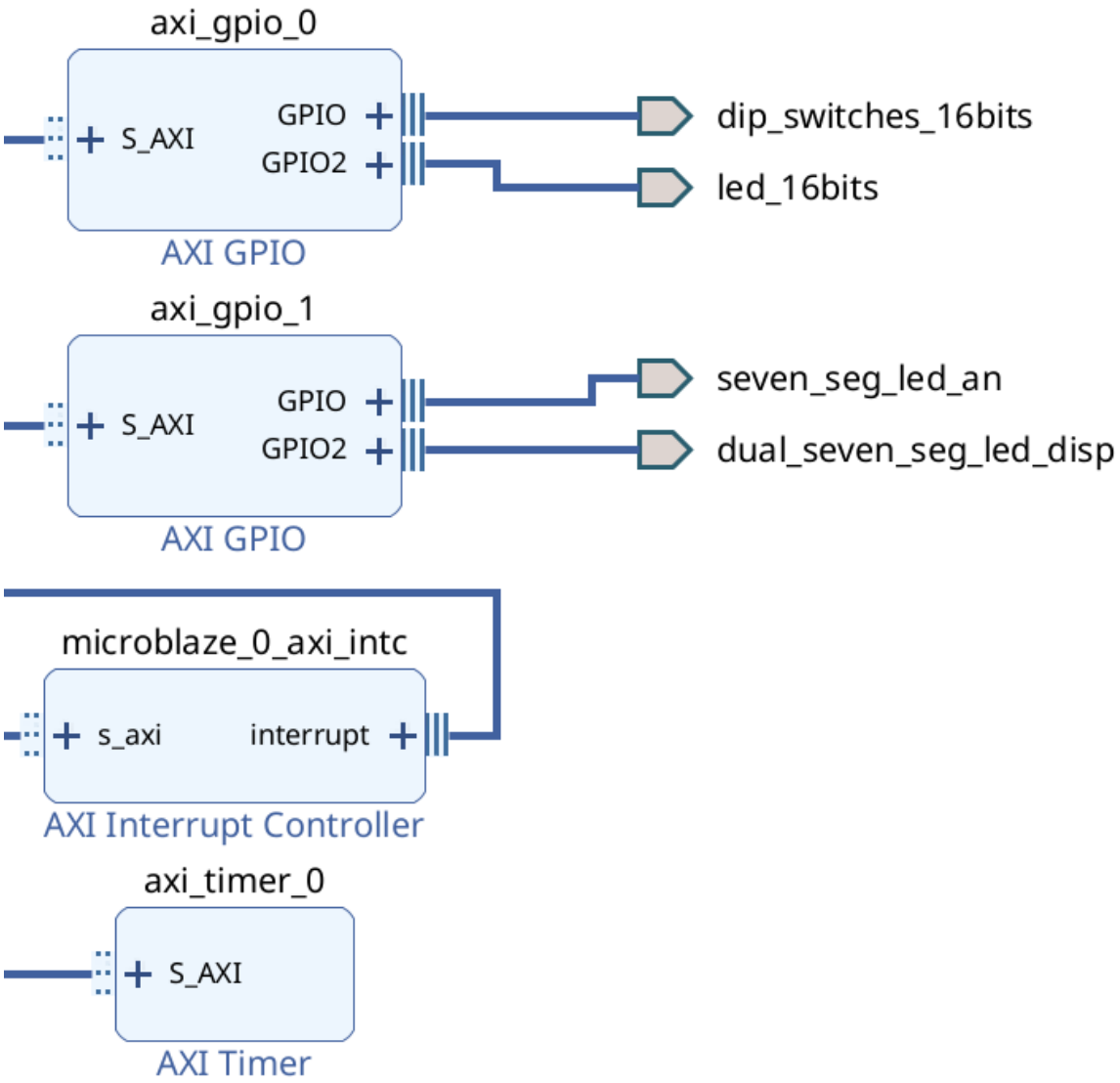

Figure 4: 硬件框图仅 IO

硬件框图基于 Nexys 4 DDR 开发板绘制，与原题略有差异，以题目为准。

GPIO_0 通道 1 连接 8 位开关，通道 2 连接 8 位 LED（高电平点亮）；GPIO_1 通道 1 连接 4 位七段数码管位选，通道 2 连接段选（低电平有效）；Timer；INTC。

其中，GPIO_0 的 ip2intr 连接 intr2，Timer 的 interrupt 连接 intr5。

各组件基地址如下：

- `GPIO_0: 0x40000000`
- `GPIO_1: 0x40010000`
- `Timer_0: 0x41C00000`
- `INTC_0: 0x41200000`

实现 LED 走马灯，数码管显示开关对应的十六进制数字（四位分别为：不显示、高位、低位、固定为 H）

挖 20 空, 数字填写十六进制:（空不全，不太记得了）

- 数码管段选（2 3 4 5）
- 位选（第四位 0xfe）
- LED GPIO 方向（TRI）
- 开关中断使能
- 中断控制器使能
- 开启定时器
- 中断服务程序逻辑（(status & 0x20)== 0x20）
- main 计数 index

```
#include <mb_interface.h>
#include <xil_io.h>
#include <xparameters.h>
#include <xgpio_l.h>
#include <xtmrctr_l.h>
#include <xintc_l.h>

void My_ISR() __attribute__((<18>));
void timer_handle();
void switch_handle();
void timer0_handle();
void timer1_handle();

#define LED_RELOAD 10000000-1
#define SEG_RELOAD 100000-1

u8 segcodes[] = {0xc0, <1>, <2>, <3>, <4>, <5>, 0x82, 0xf8, 0x80, 0x98, 0x88, 0x83,
    0xa7, 0xa1, 0x86, 0x8e};
u8 ancodes[] = {0xf7, 0xfb, 0xfd, <6>};
u8 nums[] = {0xff, 0xc0, 0xc0, <7>};
u8 switch_changed = 0;

int main() {
    Xil_Out32(0x40000004, <8>);    // 开关输入
    Xil_Out32(0x4000000C, <9>);    // LED输出
    Xil_Out32(0x40010004, 0x0);    // 共阴极
    Xil_Out32(0x4001000C, 0x0);    // 段选
    Xil_Out32(<10>, XGPIO_IR_CH1_MASK);// 中断
    Xil_Out32(0x4000011C, <11>);   // GIE

    Xil_Out32(0x41c00004, LED_RELOAD);
    Xil_Out32(0x41c00000, Xil_In32(<12>) | 0x000001f2);
    Xil_Out32(0x41c00000, Xil_In32(<12>) & ~0x00000020);

    Xil_Out32(0x41c00014, SEG_RELOAD);
    Xil_Out32(0x41c00010, Xil_In32(<13>) | 0x000001f2);
    Xil_Out32(0x41c00010, Xil_In32(<13>) & ~0x00000020);

    Xil_Out32(0x41200008, <14>);
```

```
39      Xil_Out32(0x4120001C, <15>);
40      microblaze_enable_interrupts();
41
42      while (1) {
43          if(switch_changed){
44              int switch_status = Xil_In32(0x40000000);
45              int segment_index = <16>;
46              for(int digital_index = 0; digital_index < 2; digital_index++){
47                  nums[segment_index] = segcodes[(switch_status >> (4 * digital_index)
                        ) & 0xf];
48                  segment_index --;
49              }
50              switch_changed = 0;
51          }
52      }
53  }
54
55  void My_ISR(){
56      u32 status;
57      status = Xil_In32(0x41200000);
58      if((status & 0x04) == 0x04){
59          switch_handle();
60      }
61      if(<17>)){
62          timer_handle();
63      }
64      Xil_Out32(0x4120000C, status);
65  }
66
67  void switch_handle(){
68      switch_changed = 1;
69      Xil_Out32(0x40000120, XGPIO_IR_CH1_MASK);
70  }
71
72  void timer_handle(){
73      u32 tcsr;
74      tcsr = Xil_In32(0x41c00000);
75      if((tcsr & 0x00000100) == 0x00000100){
76          timer0_handle();
77          Xil_Out32(0x41c00000, tcsr);
78      }
79
80      tcsr = Xil_In32(0x41c00010);
81      if((tcsr & 0x00000100) == 0x00000100){
82          timer1_handle();
83          Xil_Out32(0x41c00010, tcsr);
84      }
85  }
86
87  void timer0_handle(){
88      static int count = 0;
89      Xil_Out32(0x40000008, 1 << count);
90
```

```
91      count ++;
92      if(count == 8)
93          count = 0;
94  }
95
96  void timer1_handle(){
97      static int count = 0;
98      Xil_Out32(0x40010000, ancodes[count]);
99      Xil_Out32(0x40010008, nums[count]);
100
101     count ++;
102     if(count == 4)
103         count = 0;
104 }
```

6 附录

6.1 ASCII 码表

只是个码表，想不通为什么要有这么个东西。

6.2 ALU 指令

只有 ALU 指令，没有主控制器指令。

加减与或、小于设置、相等跳转

6.3 并行 IO 系统寄存器

偏移量与含义

- GPIO 的 TRI DATA TRI2 DATA2
- GPIO 的 IER ISR GIE
- TMRCTR 的 TCSR 各位
- TMRCTR 的 XTC_TIMER_COUNTER_OFFSET 等
- INTC 的 IAR ISR IER MER