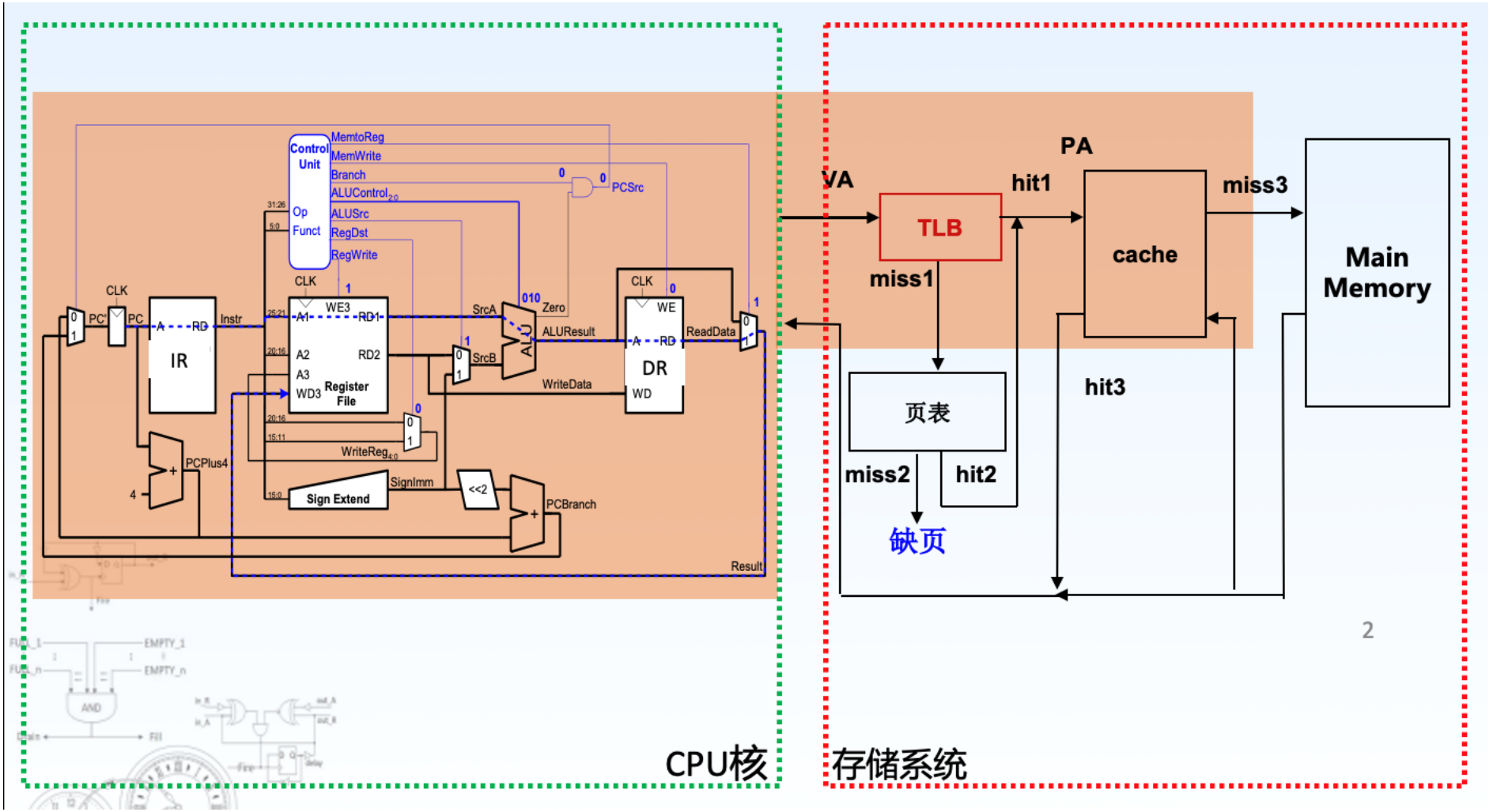
指令、CPU和层次化存储作业

Hollow Man

CPU与存储


Control Unit
MemtoReg
MemWrite
Branch
ALUControl2:0
ALUSrc
RegDst
RegWrite
Op
Funct
PCSrc
CLK
PC'
PC
IR
Instr
WE3
A1
RD1
A2
RD2
A3
WD3
Register File
SrcA
SrcB
ALU
Zero
ALUResult
WriteData
DR
WE
RD
ReadData
WriteReg4:0
PCPlus4
SignImm
Sign Extend
<<2
PCBranch
Result
CPU核
VA
TLB
hit1
miss1
PA
cache
miss3
Main Memory
页表
hit3
miss2
hit2
缺页
存储系统

高级语言到汇编程序——编译

```
sum = 0;
for (i = 0; i < 2; i++)
      sum += a[i];
*v = sum;
```

高级语言

汇编语言

0:		sum <-- 0	
1:		ap <-- A	A是数组a的起始地址
2:		i <-- 0	
3:		if (i >= 2) goto done	
4:	loop:	t <-- (ap)	数组元素a[i]的值
5:		sum <-- sum + t	累计在sum中
6:		ap <-- ap + 4	计算下个数组元素地址
7:		i <-- i + 1	
8:		if (i < n) goto loop	
9:	done:	V <-- sum	累计结果保存至地址v

分页管理——页表

- **32位系统采用分页管理地址，描述如下：**
  - 虚拟地址和物理地址均为32位，每一页的大小为4KB
  - 快表共256行，采用4路组相联映射（块表见下页）
  - 虚拟地址和物理地址均为32位，每一页的大小为4KB
  - 快表共256行，采用4路组相联映射（页表见右图）

虚页号	实页号	有效位
00000	00028	1
00001	-	0
00002	00033	1
00003	01D04	1
00004	-	0
…	…	…
0000E	A011D	1
0000F	-	0
…	…	…
00042	-	0
…	…	…
0030F	0010D	1
…	…	…

分页管理——快表

- 部分快表如下图：

行索引	第一组			第二组			第三组			第四组		
	tag	实页号	有效位	tag	实页号	有效位	tag	实页号	有效位	tag	实页号	有效位
0	0003	-	0	0009	000FD	1	0000	-	0	0700	A011D	1
1	0003	1002D	1	0108	-	0	0004	-	0	A001	-	0
2	F002	-	0	0004	00033	0	12F6	-	0	0033	-	0
3	0007	-	0	0000	01D04	1	0AAA	3BF04	1	F022	-	0
…	…	…	…	…	…	…	…	…	…	…	…	…
14	0010	012D0	1	0000	A011D	1	0700	26103	1	5001	-	0
15	000C	0010D	1	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…
63	…	…	…	…	…	…	…	…	…	…	…	…

分页管理——高速缓存

32位系统cache数据区为256KB，每一槽为16B，4路组相联。一次取32位数据

idx	第一组						第二组						第三组						第四组					
	tag	v	D0	D1	D2	D3	tag	v	D0	D1	D2	D3	tag	v	D0	D1	D2	D3	tag	v	D0	D1	D2	D3
0																								
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
58	A0 11	1	AB C0 B3 D1	1D 0D 00 C0	31 C0 E1 D4	A1 B2 20 41	00 41	1	0B B2 20 41	0A 0D 00 C0	00 D0 B3 D5	00 01 B1 1D	00 40	1	00 01 B1 1D	10 41 10 F0	00 1A 20 F5	AA 31 C0 E1	01 24	1	0B 10 41 10	0C 11 D0 B3	00 D0 D5 20	01 AB C0 B3
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1067	00 00	0	AA 20 D5 21	CA 20 11 F4	0D 10 F4 B3	F0 00 24 5F	01 0C	1	1A 20 F5 D3	0B 11 6F 32	11 D0 B3 D5	1C 41 10 F0	00 33	0	CC 0D 00 C0	AB 00 D0 B3	0A 31 C0 E1	00 CA F3 20	01 D0	1	00 2F 71 0D	01 0F 34 10	00 00 D0 1F	01 11 0F D3
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
3329	00 20	0	F1 10 11 D1	BC 00 31 11	1C 0D 00 C0	00 0D 00 C0	A0 11	1	AB 01 B1 1D	0C B2 20 41	A0 CA F3 20	AB CA F3 20	00 D0	1	00 0D 10 F4	00 31 C0 E1	00 0B B2 20	10 D0 B3 D5	00 30	0	0F 10 41 10	FF 10 D0 B3	FF 00 31 C0	FD AB 01 B1
4095	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

第1题

请使用课堂介绍过的32位MIPS/RISC-V或其他指令，将上图中的汇编语言标记为3、6和9行的语句，手工翻译成指令

- 需自行查找32个标准（通用）寄存器的功能
- 指令建议使用已经在课堂上分析过的MIPS/RISC-V基础指令（或其他基础指令）
- （在操作系统空间、程序空间，虚拟空间中）数组a的起始地址A为0x00003100，地址V的值为0x0000E2C3

设V的地址值保存在t0寄存器中，值为0x0000E2C3，sum的值存储在t1寄存器中（对应语句第零行），ap的值（数组a的起始地址值）存储在t2寄存器中（对应语句第一行），值为0x00003100，i的值存储在t3寄存器中（对应语句第二行），则：

第三行：

```
li $t4, 2 //将立即数2加载到寄存器t4中
bge $t3, $t4, done //如果 $t3 >= $t4, 则跳转到done标签处
```

第六行：

```
addi $t2, $t2, 4
```

第九行：

```
sw $t1, ($t0)
```

第2题

请根据分页管理的机制，描述虚地址0x00003XXX、0x00042XXX和0x0000EXXX，生成实地址的过程（这里大写的X表示任意的16进制数）。

由页表可知，该系统使用页号的位数为20位，则对于虚地址而言，前20位即为虚页号。快表中tag有16位。

对于虚地址0x00003XXX，其虚页号为0x00003。其前16位为0x0000，后4位的值为3，所以应该在行索引3的4个组中寻找tag为0000的实页号。最终第二组的tag号符合，且有效位为1，所以得到实页号0x01D04，则实地址为0x01D04XXX。

对于虚地址0x00042XXX，其虚页号为0x00042。其前16位为0x0004，后4位的值为2，所以应该在行索引2的4个组中寻找tag为0004的实页号。最终第二组的tag号符合，但是有效位为0，发生了快表缺失，因而需要到页表中查找。页表中查到虚页号为0x00042的有效位为0，因而发生了缺页，需要到磁盘中找。

对于虚地址0x0000EXXX，其虚页号为0x0000E。其前16位为0x0000，后4位的值为14，所以应该在行索引14的4个组中寻找tag为0000的实页号。最终第二组的tag号符合，且有效位为1，所以得到实页号0xA011D，则实地址为0xA011DXXX。

第3题

请分析物理地址0x01D042B7，0x004003A2和0xA011D01B所对应的32位数据。

MIPS为大端存储，cache组地址:256KB/(4*16B)=4k=2^12,
因而cache组地址12位
cache块内地址:16=2^4,cache块内地址4位

对于物理地址0x01D042B7，其tag号为0x01D0，组地址为0x42B，转换为十进制为1067，块内地址为7。则在第四组中找到tag号01D0，因为块内地址为7，向高位读取32位数据，得到数据为0x100000D0

对于物理地址0x004003A2，其tag号为0x0040，组地址为0x03A，转换为十进制为58，块内地址为2。则在第三组中找到tag号0040，因为块内地址为2，向高位读取32位数据，得到数据为0xB11D1041

对于物理地址0xA011D01B，其tag号为0xA011，组地址为0xD01，转换为十进制为3329，块内地址为11。则在第三组中找到tag号A011，因为块内地址为11，向高位读取32位数据，得到数据为0x20ABCAF3

第4题

参考第2、3题，请分析第1题中MIPS指令的执行过程，（可画图）描述指令的各个执行阶段。

（下述过程略去取指和译码这两个共有阶段，即首先指令被取出，在指令被取出的过程中首先通过PC寄存器获取到指令的物理地址，然后访问cache，命中则取指成功，未命中则访问内存取指。然后PC自加+4。指令译码器按照预定的指令格式，对取回的指令进行拆分和解释，识别区分出不同的指令类别以及各种获取操作数的方法，更改控制单元的值。）

li $t4, 2：将值2写入寄存器t4中

bge $t3, $t4, done：将寄存器t3的值和寄存器t4的送入ALU，op设置为做减法，如果输出的标志位中负数标志位为0，则PC的值加当期PC值到done标记的地址偏移量，从而跳转到done处。

addi $t2, $t2, 4：将寄存器t2的值和立即数2送入ALU，op设置为做加法，输出的结果写回寄存器t2中。

sw $t1, ($t0)：读取出寄存器t1中的值，随后读取出寄存器中t0的值，为0x0000E2C3，其虚页号为0x0000E。其前16位为0x0000，后4位的值为14，所以应该在行索引14的4个组中寻找tag为0000的实页号。最终第二组的tag号符合，且有效位为1，所以得到实页号0xA011D，则实地址为0xA011E2C3。又因为该实地址tag号为0xA011，组地址为0xE2C，转换为十进制为3628，只需要寻找idx为3628的那一组地址中tag为A011的那一块。如果找到则将寄存器t1中的值写入，否则从内存中寻找，或者将该地址值装入。最后写入内存。