题号：A1　Allocation (10 points)

1. [1]　33 (2’)　[2]　48 (2’) (56 also give 2 points)
2. [3]　33 (2’)　[4]　24 (2’) (32 also give 2 points)
3. [5]　The malloc will call sbrk to extend the heap. (2’)

题号：A2　Replacement Policy (10 points)

1. 一个空给 1 分

Reference String	1	4	5	3	1	5	1	8
Device Content	1	1	1	3	3	3	3	8
	-	4	4	4	1	1	1	1
	-	-	5	5	5	5	5	5

2. 一个空给 0.5 分

Reference String	1	4	5	3	1	5	1	8
Device Content	1	1	1*	3	3	3	3	3*
	*	4	4	4*	1	1	1	1
	-	*	5	5	5*	5*	5*	8
Reference Bit	1	1	1	1	1	1	1	0
	0	1	1	0	1	1	1	0
	0	0	1	0	0	1	1	1

题号：B1　Cache (14 points)

1. [1] 58　[2] 3　[3] 3
2. 1/2

3. 4*（1/2）+(200+4)*（1/2）=104 cycles (102 也给了 2 分， 其他能看出来理解 average access time 是什么的答案也给了 1 分)

4. i,j,k or j,i,k.

Cache miss rate: 1/4

题号：B2 Memory (14 points)

1. [1] 48 [2] 16

第二题，PARENT CHILD 写反了的答案给了全分，写了四个 bytes 的答案给了全分，写成十六进制的答案给了全分。

2. [3] 5 [4] 2 [5] 10 [6] 2

写成十六进制的答案给了全分。

3. 4

题号：C1　Optimization (26 points)

1.

```
void conv1_ans1(struct data *signal, struct data *kernel, double *result,
           size_t *len)
{
    /* Reducing Procedure Calls */ (4’)
    double *signal_data = get_data_start(signal);
    size_t signal_len = get_data_len(signal);
    double *kernel_data = get_data_start(kernel);
    size_t kernel_len = get_data_len(kernel);
    size_t i = 0, j = 0, kmin = 0, kmax = 0, limit;
    int result_len = signal_len + kernel_len - 1;
    double acc0, acc1;

    /* Eliminating Unneeded Memory References */ (4’)
    *len = result_len;
    /* Eliminating Loop Inefficiencies */ (4’)
    for (i = 0; i < result_len; i++)
    {
        acc0 = acc1 = 0;
        kmin = (i >= kernel_len - 1) ? i - (kernel_len - 1) : 0;
        kmax = (i < signal_len - 1) ? i + 1 : signal_len;
        limit = kmax - 1;
        /* Loop Unrolling */ (4’)
        for (j = kmin; j < limit; j+=2) {
            /* Multiple Accumulators */
            acc0 += signal_data[j] * kernel_data[i - j];
            acc1 += signal_data[j + 1] * kernel_data[i - j - 1];
        }
        for (; j < kmax; j ++) {
            acc0 += signal_data[j] * kernel_data[i - j];
        }
        result[i] = acc0 + acc1;
    }
}
```

只要意思对了，每个优化就给 4 分

2.

When there is no store/load dependencie, aka, result has no relevance with the signal or kernel functions, the **combine** will have the minimal CPE, which is 1 in this case. Only the integer add will be on the critical path. (5’) (Right condition explaination will give 4 points)

When the result pointer points to the next element of signal or kernel (e.g. result = signal->data + 1), the CPE is 11. In this case, the load of one iteration depends on the result of the store from the previous iteration. The critical path is the "load-multiply-store" (5')

只要 CPE 中有 11 以及 1 的，可以分别给一半的分，虽然没解释对。

题号：C2 Address Translation (26 points)

1. [1] 128 [2] 10 [3] 24 [4] 20 [5] 12

2. [6] 37e [7] N [8] -- [9] --
   [10] f2 [11] Y [12] N [13] b43f3

题号：D1 Relocation (21 points)

1. [1] 4 [2] OBJECT [3] GLOBAL [4] COM
   [5] 00000000 [6] 0 [7] NOTYPE [8] UND
   [9] 0 [10] NOTYPE [11] GLOBAL [12] UND
2. [1] foo [2] -4
   [3] R_X86_64_PC32 [4] a [5] -4
   [6] R_X86_64_PC32 [7] len [8] -4
   [9] R_X86_64_PC32 [10] bar [11] -4
   [12] R_X86_64_PC32 [13] b [14] -4
3. [1] 4f 00 00 00 [2] 88 09 20 00
   [3] 29 09 20 00 [4] 1e 09 20 00

题号：D2 Relocation (21 points)

1. [1] 4 [2] OBJECT [3] GLOBAL [4] COM
   [5] 00000000 [6] 0 [7] NOTYPE [8] UND
   [9] 0 [10] NOTYPE [11] GLOBAL [12] UND
2. [1] foo [2] -4
   [3] R_X86_64_PC32 [4] b [5] -4

[6] R_X86_64_PC32 [7] len [8] -4

[9] R_X86_64_PC32 [10] bar [11] -4

[12] R_X86_64_PC32 [13] b [14] -4

3. [1] 53 00 00 00 [2] 88 09 20 00

[3] 19 09 20 00 [4] 27 09 20 00

题号：E1 CPU (29 points)

1. [1]
```
icode:ifun <- M1[PC]
rA:rB <- M1[PC+1]
valC <- M8[PC+2]
valP <- PC+10
```

[2] `valA <- R[rA]`

[3] `cnd <- cond(CC,ifun)` [4] --

[5] -- [6] `PC <- cnd ? valC : valA`

2. [1] `E_icode == ISSJXX && !e_cnd`

[2] -- [3] bubble [4] bubble [5] --[6] --

3.

M	[1] --
E	[2] ssjxx
D	[3] --

M	[4] --
E	[5] --
D	[6] ret

Condition	Pipeline register				
	F	D	E	M	W
ssjxx	normal	bubble	bubble	normal	normal
ret	stall	bubble	normal	normal	normal
combination	stall	bubble	bubble	normal	normal

4. [1] 0.04 [2] 0.08 [3] 0.03 [4] 2 [5] 0.16

[6] 1.31

5. The origin design is better. The new design cannot take advantage of branch prediction. Because both address need to be read at DECODE period, it still need stall one cycle even though prediction is correct.

题号：E2　CPU (29 points)

1. [1] `icode:ifun <- M1[PC]`
   `rA:rB <- M1[PC+1]`
   `valC <- M8[PC+2]`
   `valP <- PC+10`

   [2] `valA <- R[rA]`

   [3] `cnd <- cond(CC,ifun)`　　[4] `--`

   [5] `--`　　[6] `PC <- cnd ? valC : valA`

2. [1] `E_icode == ISSJXX && !e_cnd`

   [2] `--`　　[3] `bubble` [4] `bubble` [5] `--`[6] `--`

3.

M	[1] --
E	[2] ssjxx
D	[3] --

M	[4] --
E	[5] --
D	[6] ret

Condition	Pipeline register				
	F	D	E	M	W
ret	stall	bubble	normal	normal	normal
ssjxx	normal	bubble	bubble	normal	normal
combination	stall	bubble	bubble	normal	normal

4. [1] 0.03　　[2] 0.16　　[3] 0.03　　[4] 2　　[5] 0.12

   [6] 1.34

5. The origin design is better. The new design cannot take advantage of branch prediction. Because both address need to be read at DECODE period, it still need stall one cycle even though prediction is correct.