Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg

# Microprocessors 3

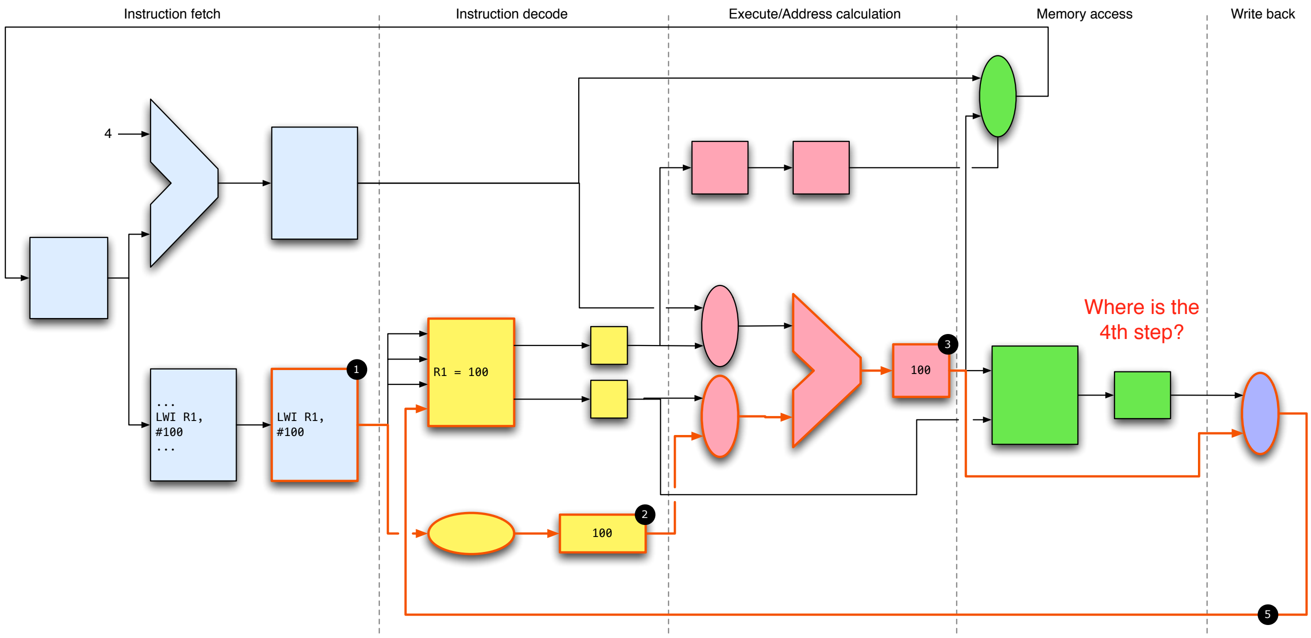
## Report lab 3 : Pipeline

Jonathan Stoppani et Elias Medawar

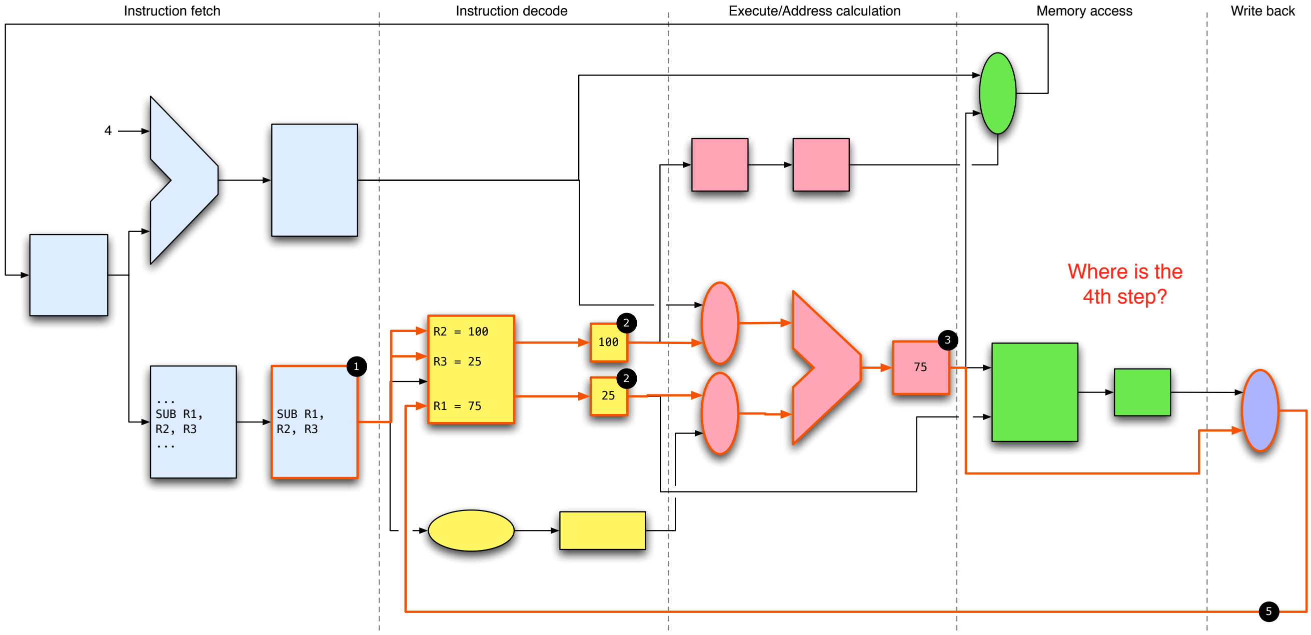
Version: March 9, 2011

# 1 Separated instructions, DLX processor

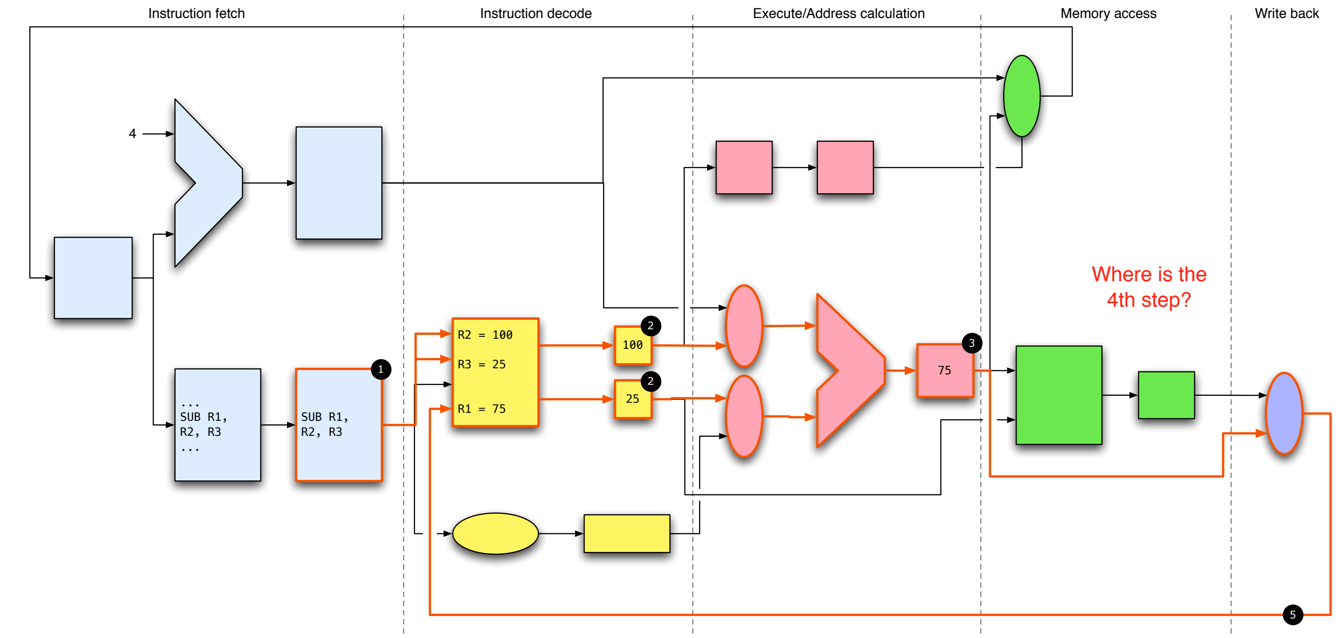
## 1.1 LWI R1 , #100



## 1.2 SUB R1 , R2, R3

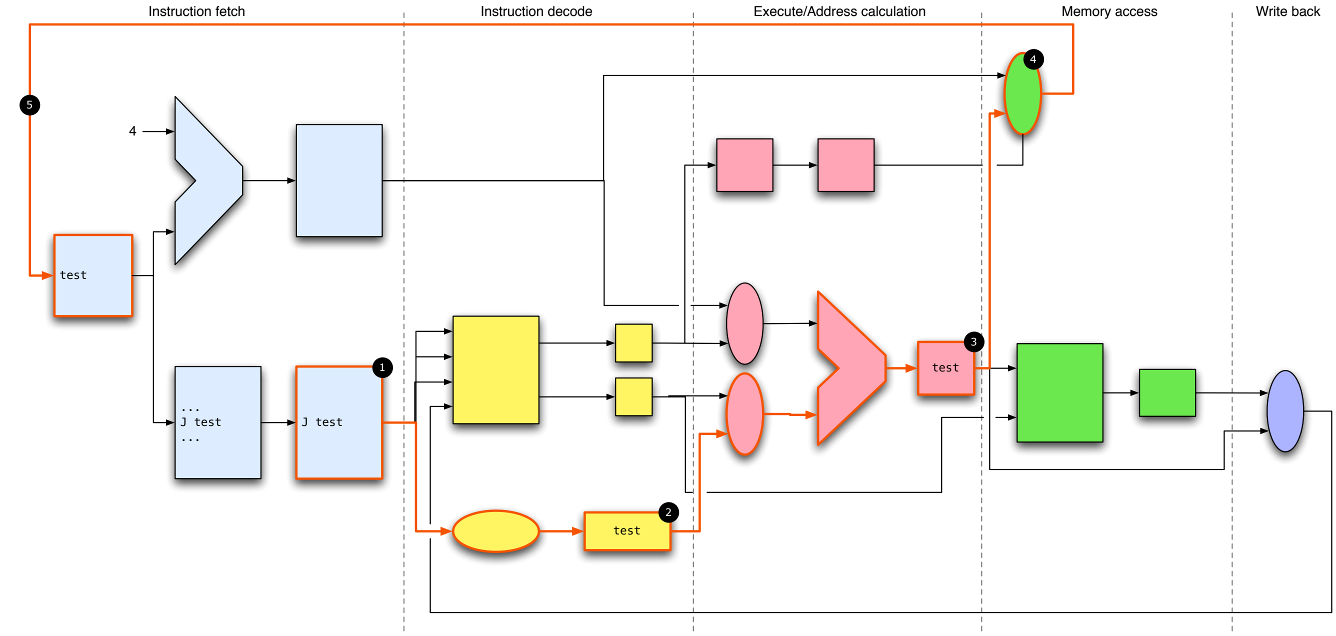


## 1.3 SUB R1 , R2, R3

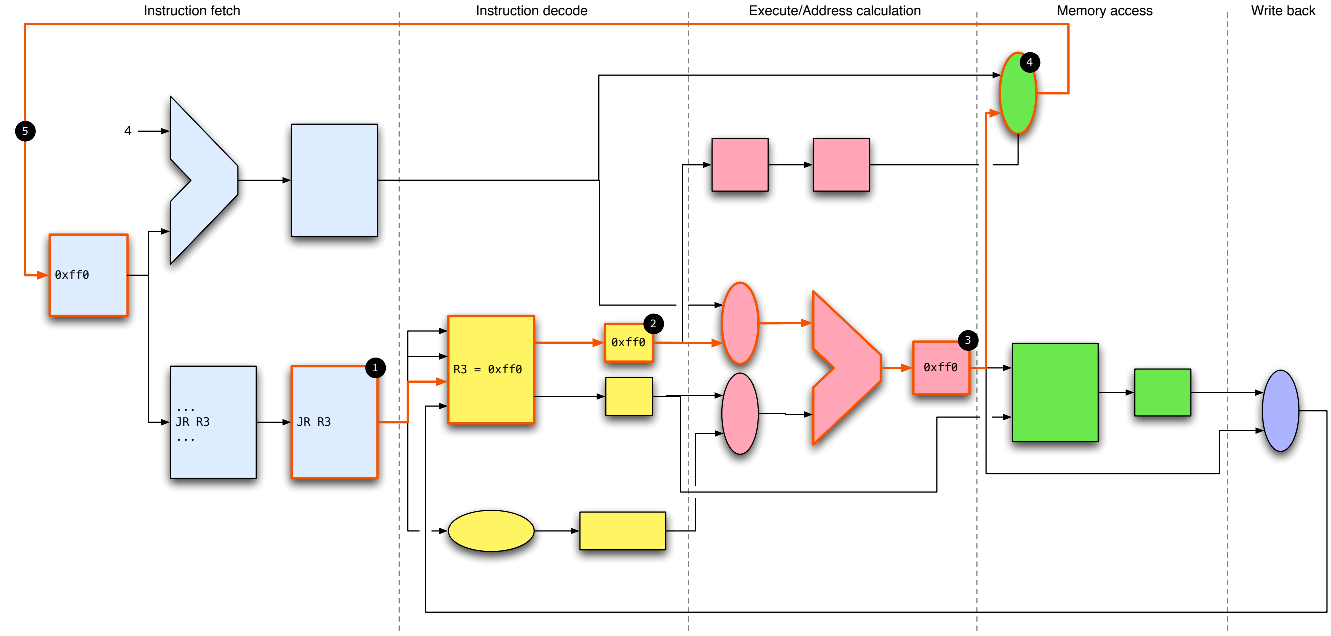


Where is the 4th step?

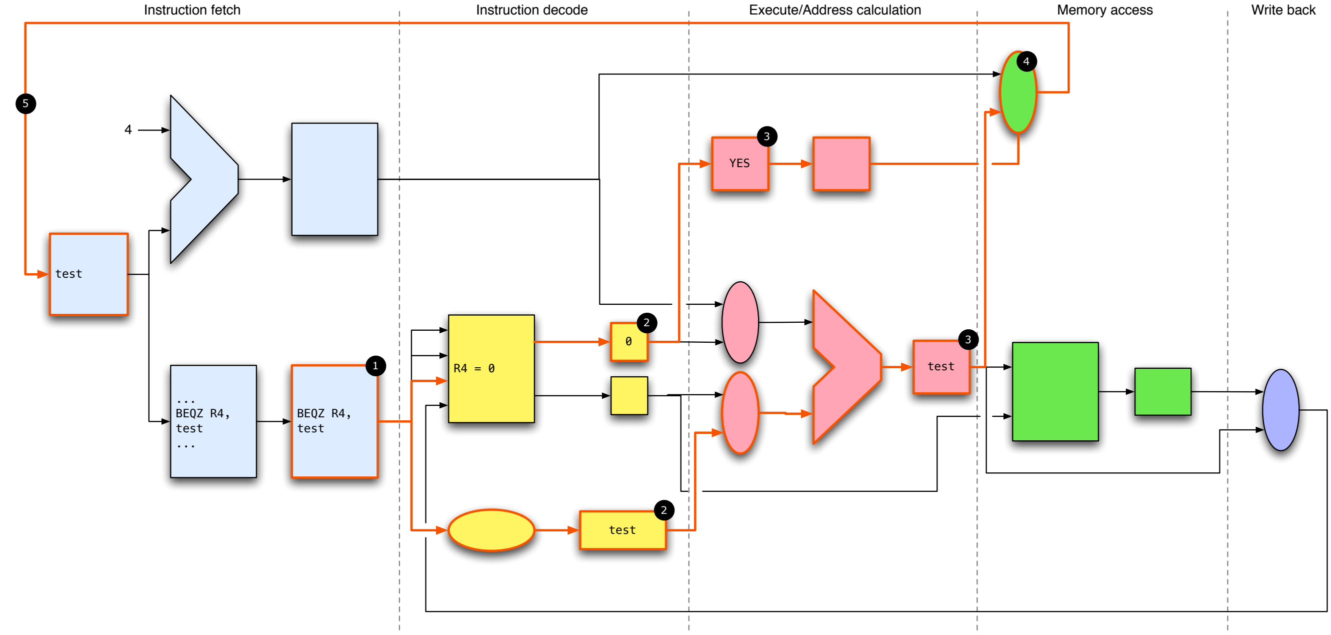
## 1.4 J test



## 1.5 JR R3



## 1.6 BEQZ R4, test

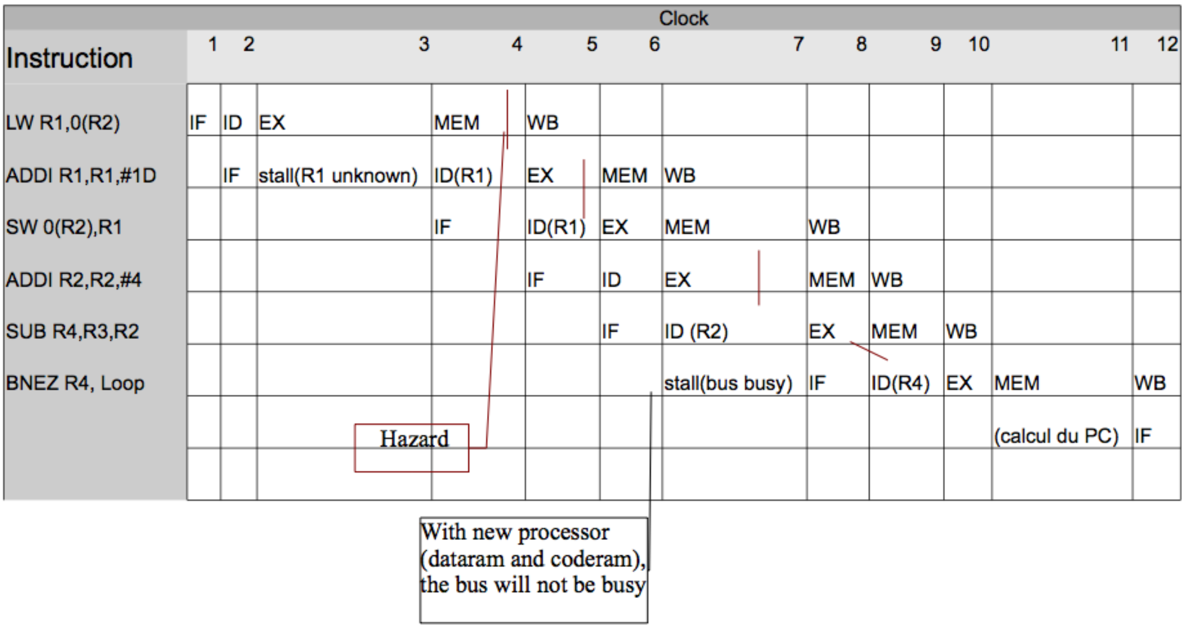


## 1.7 ADD R1, R2 ,2(R3)

This instruction is not feasible with the architecture(load-store) of this processor. This architecture is allow not to load from memory directly to the ALU registers

# 2 Program, DLX processor

## 2.1 DLX path and hazards



| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LW R1,0(R2) | IF | ID | EX | MEM | WB | | | | | | | |
| ADDI R1,R1,#1D | | IF | stall(R1 unknown) | ID(R1) | EX | MEM | WB | | | | | |
| SW 0(R2),R1 | | | | IF | ID(R1) | EX | MEM | WB | | | | |
| ADDI R2,R2,#4 | | | | | IF | ID | EX | MEM | WB | | | |
| SUB R4,R3,R2 | | | | | | IF | ID (R2) | EX | MEM | WB | | |
| BNEZ R4, Loop | | | | | | | stall(bus busy) | IF | ID(R4) | EX | MEM | WB |
| | | | | | | | | | | | (calcul du PC) | IF |
| | | | | | | | | | | | | |

Clock
Hazard
With new processor (dataram and coderam), the bus will not be busy

The hazards are drawn in red.

# 3 Program, Intel Processor

## 3.1 Time measurement

```c
static __inline__ unsigned long long rdtsc(void)
{
  unsigned long long int x;
     __asm__ volatile (".byte 0x0f, 0x31" : "=A" (x));
     return x;
}
```

We have measured the time with this function and the proposed method of this curse .

## 3.2 Initial program

```c
int main(int argc, char * argv[]) {
   int arraySize = atoi(argv[1]);\/\/ The size of the buffer array

   int buffer[arraySize];

   int ret,j,sum;
   long long t1, t2, t1ms;
   int which = PRIO_PROCESS;
   id_t pid;
   int oldPriority;
   int priority;

   pid = getpid();
   oldPriority = getpriority(which, pid);
   priority = -20;
   ret = setpriority(which, pid, priority);

   t1 = rdtsc();
   sleep(1);
   t2 = rdtsc();

```

```
    t1ms = (t2 - t1) / 1000L;

    t1 = rdtsc();

    for (j = 0; j < arraySize; j++) {
        sum+=buffer[j];
    }

    t2 = rdtsc();

    printf(" - Initial:    delta t = %lld [ms]\n", (t2 - t1) / t1ms);

    ret = setpriority(which, pid, oldPriority);
    return 0;
}
```

## 3.3 First optimisation program

```
int main(int argc, char * argv[]) {
    int arraySize = atoi(argv[1]);\/\/ The size of the buffer array

    int buffer[arraySize];

    int ret,j,sum,a,b,c,d;
    long long t1, t2, t1ms;
    int which = PRIO_PROCESS;
    id_t pid;
    int oldPriority;
    int priority;

    pid = getpid();
    oldPriority = getpriority(which, pid);
    priority = -20;
    ret = setpriority(which, pid, priority);

    t1 = rdtsc();
    sleep(1);
    t2 = rdtsc();

    t1ms = (t2 - t1) / 1000L;

    t1 = rdtsc();

    for (a=0,b=0,c=0,d=0,j = 0; j < arraySize; j+=4) {
        a+=buffer[j];
        b+=buffer[j+1];
        c+=buffer[j+2];
        d+=buffer[j+3];
    }
    sum = a+b+c+d;
    t2 = rdtsc();

    printf(" - firstOptimisation:    delta t = %lld [ms]\n", (t2 - t1) / t1ms);

    ret = setpriority(which, pid, oldPriority);
    return 0;
}
```

## 3.4 Second optimisation program

```
int main(int argc, char * argv[]) {
    int arraySize = atoi(argv[1]);\/\/ The size of the buffer array
    int buffer[arraySize];
    int ret,j,sum,a,b,c,d,e,f,g,h;
    long long t1, t2, t1ms;
    int which = PRIO_PROCESS;
    id_t pid;
    int oldPriority;
    int priority;

    pid = getpid();
    oldPriority = getpriority(which, pid);
    priority = -20;
    ret = setpriority(which, pid, priority);

    t1 = rdtsc();
    sleep(1);
    t2 = rdtsc();

    t1ms = (t2 - t1) / 1000L;

    t1 = rdtsc();

    for (a=0,b=0,c=0,d=0,e=0,f=0,g=0,j = 0; j < arraySize; j+=7) {
        a+=buffer[j];
        b+=buffer[j+1];
        c+=buffer[j+2];
        d+=buffer[j+3];
        e+=buffer[j+4];
        f+=buffer[j+5];
        g+=buffer[j+6];
    }
    sum = a+b+c+d+e+f+g;
    t2 = rdtsc();

    printf(" - secondOptimisation:   delta t = %lld [ms]\n", (t2 - t1) / t1ms);

    ret = setpriority(which, pid, oldPriority);
    return 0;
}
```

## 3.5 Third optimisation program

```

int main(int argc, char * argv[]) {
     int arraySize = atoi(argv[1]);\/\/ The size of the buffer array

    int buffer[arraySize];

    int ret,j,sum,a,b,c,d,e,f,g,h;
    long long t1, t2, t1ms;
    int which = PRIO_PROCESS;
    id_t pid;
    int oldPriority;
    int priority;

    pid = getpid();
    oldPriority = getpriority(which, pid);
    priority = -20;
    ret = setpriority(which, pid, priority);

    t1 = rdtsc();
    sleep(1);
    t2 = rdtsc();

    t1ms = (t2 - t1) / 1000L;

    t1 = rdtsc();
    for (a=0,b=0,c=0,d=0,e=0,f=0,g=0,j = 0; j < arraySize; j+=29) {
        a+=buffer[j];
        b+=buffer[j+1];
        c+=buffer[j+2];
        d+=buffer[j+3];
        e+=buffer[j+4];
        f+=buffer[j+5];

        g+=buffer[j+6];
        a+=buffer[j+7];
        b+=buffer[j+8];
        c+=buffer[j+9];
        d+=buffer[j+10];
        e+=buffer[j+11];
        f+=buffer[j+12];
        g+=buffer[j+13];

        a+=buffer[j+14];
        b+=buffer[j+15];
        c+=buffer[j+16];
        d+=buffer[j+17];
        e+=buffer[j+18];
        f+=buffer[j+19];
        g+=buffer[j+21];

        a+=buffer[j+22];
        b+=buffer[j+23];
        c+=buffer[j+24];
        d+=buffer[j+25];
        e+=buffer[j+26];
        f+=buffer[j+27];
        g+=buffer[j+28];
    }

    sum = a+b+c+d+e+f+g;
    t2 = rdtsc();

    printf(" - thirdOptimisation:     delta t = %lld [ms]\n", (t2 - t1) / t1ms);

    ret = setpriority(which, pid, oldPriority);
    return 0;
}
```

## 3.6 MakeFile

```

ARRAY_SIZE=400000000

initial: initial.c
    gcc initial.c -o initial

firstOptimisation: firstOptimisation.c
    gcc firstOptimisation.c -o firstOptimisation

secondOptimisation: secondOptimisation.c
    gcc secondOptimisation.c -o secondOptimisation

thirdOptimisation: thirdOptimisation.c
    gcc thirdOptimisation.c -o thirdOptimisation

timeit: initial firstOptimisation secondOptimisation thirdOptimisation
    @echo Timing a array for an array of $(ARRAY_SIZE) int:
    @./initial $(ARRAY_SIZE)
    @./firstOptimisation $(ARRAY_SIZE)
    @./secondOptimisation $(ARRAY_SIZE)
    @./thirdOptimisation $(ARRAY_SIZE)

clean:
    rm -f initial firstOptimisation secondOptimisation thirdOptimisation
```

## 3.7 Result

```
    Timing a the execution for an array of 400000000 int:
     - Initial:    delta t = 2216 [ms]
     - firstOptimisation:    delta t = 1340 [ms]
     - secondOptimisation:   delta t = 1288 [ms]
     - thirdOptimisation:    delta t = 1084 [ms]
```