1093319-魏博彥 project3 說明文件

- Input:

  老師網站上面一模一樣，然後每個 instruction 要執行幾個 cycle 我是照著影片裏面的寫，也就是 ADD、SUB 是 2，MUL 是 10，DIV 是 40。

```
ADDI F1, F2, 1
SUB F1, F3, F4
DIV F1, F2, F3
MUL F2, F3, F4
ADD F2, F4, F2
ADDI F4, F1, 2
MUL F5, F5, F5
ADD F1, F4, F4
```

- 我自己的手算模擬:

※ 我不管怎麼算，最後面結束的 cycle 都是在 64 而不是在 63 (老師給的範列答案最後一個是 63) ，但其他都一樣，所以我是先照著我的這張表去對的。

- Output 範列有另外附上。

- 我有三個 struct 分別用來宣告我的 input (data)、RS、buffer。

```
typedef struct A//data
{
    string Reg;
    int Fa;
    int Fb;
    int Fc;
}A;

typedef struct B//rs
{
    char Operator;
    int b;
    int c;
    bool ready_all = false;//true means ready
    bool ready_b = false;
    bool ready_c = false;
    bool empty = true;//true means empty
}B;

typedef struct C//buffer
{
    int rs;
    int b;
    char opert;
    int c;
    int cycl;//紀錄run幾個cycle後可以wr
}C;
```

```
int RF[6] = { -1,0,2,4,6,8 };//第一個-1不要用
vector<int> RAT[5] = {};//始終指每項只能有一個數字
//vector<stack<int>>RAT[5];
B RS_add[3] = {};
B RS_mul[2] = {};
int cycle = 0;
const int add_cycle = 2, mul_cycle = 10, div_cycle = 40;
bool unit_add = true;//ALU unit//true means empty
bool unit_mul = true;
C buffer_add = {};
C buffer_mul = {};
bool out = false;//決定output與否
int RS_posa, RS_posm;//紀錄現在正在unit執行的RS為RS幾
//防止遞迴呼叫時多做
bool checkdis_a = false;//true時表示再次呼叫 但總共只讓他做一次
bool checkdis_m = false;
```

- Main 裡面讀檔後將資料放入 data queue，然後就呼叫 Tomasulo 開始模擬。

```
    Tomasulo(Data);

    infile.close();
    return 0;
}
```

- Tomasulo:

```
void Tomasulo(queue<A>& data)
{

    bool  first = true;//第一次的話如果可以dispatch要在下一個cycle才可以做

    while (!data.empty())
    {
        cycle++;

        if (!first)dispatch();
        else first = false;
        A u = data.front();
        //cout << u.Reg << " " << u.Fa << "," << u.Fb << "," << u.Fc << endl;
        if (issue(u))
        {
            data.pop();//issue sucessful
            out = true;//out有變化就要output
        }

        output();
        out = false;
    }

    while (!unit_add || !unit_mul)//unit還有東西所以要繼續算下去
    {
        cycle++;
        dispatch();
        output();
        out = false;
    }
}
```

第一次進來的話直接 issue 就好，第一次以外的我會先做 dispatch 再 issue 因為我 cpature 會在 dispatch 裡呼叫 write_back 做，然後 issue 成功的時候 inst.會 pop 掉。這整組的循環會做到 data 為空為止，即為沒東西可以 issue 時，我再接續著做 dispatch 直到所有 operator unit 都閒置下來，結束。

- Issue:

```
bool issue(A inst)
{
    bool find_empty = false;
    int whrs;//where is rs

    //RS有空位才可以進
    if (inst.Reg == "ADDI" || inst.Reg == "ADD" || inst.Reg == "SUB")
    {
        for (int i = 0; i < 3; i++)//從第一個開始找有空的RS
        {
            if (RS_add[i].empty)// RSi ----> ex:i=0 ,表RS1//要加1
            {
                RS_add[i].empty = false;//更新為不為空
                find_empty = true;//RS有位子可以issue
                whrs = i + 1;

                // operator: +
                if (inst.Reg == "ADDI" || inst.Reg == "ADD")RS_add[i].Operator = '+';
                // operator: -
                else RS_add[i].Operator = '-';

                //check RAT
                /******************check inst.b 在RAT中有無東西**********************/
                if (RAT[inst.Fb - 1].empty())//RAT沒東西
                {
                    RS_add[i].b = RF[inst.Fb];//則Fx值為RF裡的值
                    //RAT[inst.Fa - 1].push_back(i + 1);//RAT給予對應的RSi//inst.Fa表示為Fx//ex: inst.Fa=1  表F1
                    RS_add[i].ready_b = true;//RS.b is already
                }
                else//RAT不為空//要等待別人write back
                {
                    //RS_add is not ready
                    RS_add[i].ready_b = false;
                    RS_add[i].b = RAT[inst.Fb - 1][0];//給予RS編號
```

```
                RS_add[i].b = RAT[inst.Fb - 1][0];//給予RS編號
                //RAT[inst.Fa - 1].push_back(i + 1);//RAT給予對應的RSi//inst.Fa表示為Fx//ex: inst.Fa=1  表F1
            }

            /*****************check inst.c 在RAT中有無東西**********************/
            if (inst.Reg == "ADDI")
            {
                RS_add[i].c = inst.Fc;//addi , so inst.Fc is a number//ready is already true
                RS_add[i].ready_c = true;
            }
            else//add && sub
            {
                if (RAT[inst.Fc - 1].empty())//RAT沒東西
                {
                    RS_add[i].c = RF[inst.Fc];//則Fx值為RF裡的值
                    RS_add[i].ready_c = true;
                }
                else//RAT不為空//要等待別人write back
                {
                    //RS_add is not ready
                    RS_add[i].ready_c = false;//就算RS.b is true但這邊RS.c還是false
                    RS_add[i].c = RAT[inst.Fc - 1][0];//給予RS編號
                }
            }

            /***********inst.a 在RAT中有無RS************************************/
            if (!RAT[inst.Fa - 1].empty())RAT[inst.Fa - 1].clear();//若RAT已有RS則先清掉在新增
            RAT[inst.Fa - 1].push_back(i + 1);//RAT給予對應的RSi//inst.Fa表示為Fx//ex: inst.Fa=1  表F1

            /***更新整體ready sate***/
            if (RS_add[i].ready_b && RS_add[i].ready_c)RS_add[i].ready_all = true;//already for dispatch
            break;
        }
    }
```

```
    }
}
else//MUL DIV ----> RS編號+3
{
    for (int i = 0; i < 2; i++)//從第一個開始找有空的RS
    {
        if (RS_mul[i].empty)
        {
            RS_mul[i].empty = false;//更新為不為空
            find_empty = true;//RS有位子可以issue
            whrs = i + 1;

            // operator: +
            if (inst.Reg == "MUL")RS_mul[i].Operator = '*';
            // operator: /
            else RS_mul[i].Operator = '/';

            //check RAT
            /*****************check inst.b 在RAT中有無東西*************************/
            if (RAT[inst.Fb - 1].empty())//RAT沒東西
            {
                RS_mul[i].b = RF[inst.Fb];//則Fx值為RF裡的值
                RS_mul[i].ready_b = true;//RS.b is already
            }
            else//RAT不為空//要等待別人write back
            {
                RS_mul[i].ready_b = false;
                RS_mul[i].b = RAT[inst.Fb - 1][0];//給予RS編號
            }

            /*****************check inst.c 在RAT中有無東西*************************/
            if (RAT[inst.Fc - 1].empty())//RAT沒東西
            {
                RS_mul[i].c = RF[inst.Fc];//則Fx值為RF裡的值
```

```
                    RS_mul[i].c = RF[inst.Fc];//則Fx值為RF裡的值
                    RS_mul[i].ready_c = true;
                }
                else//RAT不為空//要等待別人write back
                {
                    RS_mul[i].ready_c = false;//就算RS.b is true但這邊RS.c還是false
                    RS_mul[i].c = RAT[inst.Fc - 1][0];//給予RS編號
                }

                /***********inst.a 在RAT中有無RS**************************************/
                if (!RAT[inst.Fa - 1].empty())RAT[inst.Fa - 1].clear();//若RAT已有RS則先清掉在新增
                RAT[inst.Fa - 1].push_back(i + 4);//RAT給予對應的RSi//inst.Fa表示為Fx//ex: inst.Fa=1  表F1

                /***更新整體ready sate***/
                if (RS_mul[i].ready_b && RS_mul[i].ready_c)RS_mul[i].ready_all = true;//already for dispatch
                break;
            }
        }
    }

    return find_empty;
}
```

主要想法就是:

1.IQ 進 instruction 到 RS。

2.check RAT 中有無紀錄，沒有的話值就是 RF 的值，有的話就是 RS 未知數。

3.RAT 裡對應的 F 給予 RS 編號，如果已有編號->取代

4.IQ pop

細部操作請洽註解，然後我這會回傳一個 bool 值，true 代表 issue 成功，所以 IQ (就是我的 data) 會 pop( )，false 則相反。

- Dispatch:

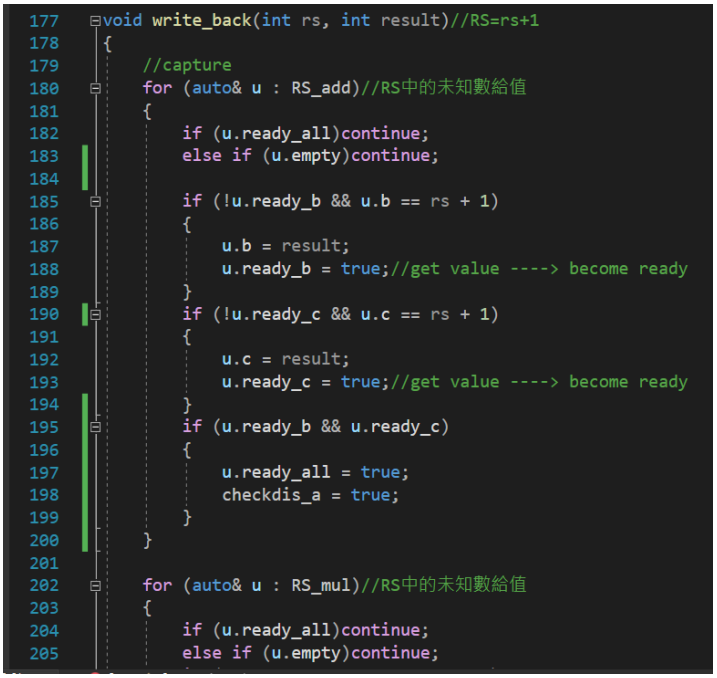
分兩個 unit 做，分別是 ADD 的與 MUL 的，然後我只要 unit 是空的，我就會去 RS 裡面找 ready 的做，然後放入 buffer 裡，並給予對應運算所要花的 cycle 是多少。在我 unit 裡面有東西的時候我先檢查他的剩餘 cycle，如果不為 0 則表這個還沒算完，所以繼續 -1；為 0 的時候，代表運算完成，所以可以做 write back 了，並釋放 RS 與 unit。

```
void dispatch()
{
    if (unit_add && checkdis_a)checkdis_a = false;
    else if (unit_add)//add unit is empty
    {
        for (int i = 0; i < 3; i++)
        {
            if (RS_add[i].ready_all)//選一個ready進入buffer
            {
                out = true;
                RS_posa = i;
                buffer_add.rs = i;
                unit_add = false;
                buffer_add.opert = RS_add[i].Operator;
                buffer_add.b = RS_add[i].b;
                buffer_add.c = RS_add[i].c;
                buffer_add.cycl = 2 - 1;//假設ADD ADDi SUB 均要做2 cycle
                break;
            }
        }
    }
    else
    {
        if (buffer_add.cycl == 0)//可以wr
        {
            out = true;
            int result;
            if (buffer_add.opert == '+')result = buffer_add.b + buffer_add.c;//count the result
            else result = buffer_add.b - buffer_add.c;
```

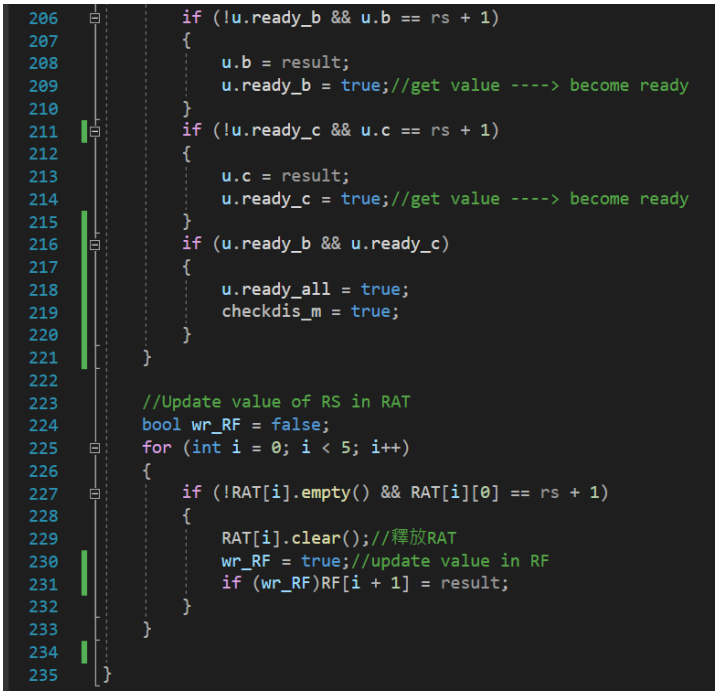
```
265             else result = buffer_add.b - buffer_add.c;
266
267             RS_add[RS_posa] = {};//RS釋放
268             write_back(buffer_add.rs, result);//wr to that rs
269             buffer_add = {};//buffer become empty
270             unit_add = true;//become empty
271             checkdis_m = true;
272             dispatch();//unit空間釋出 所以可以找一個ready的作dispatch//直接地回會多檢一次1
273         }
274         else buffer_add.cycl--;
275     }
276
277     /*****************************************************/
278     if (checkdis_m)
279     {
280         checkdis_m = false;
281         return;
282     }
283     else if (unit_mul && checkdis_m)checkdis_m = false;
284     else if (unit_mul)//mul unit is empty
285     {
286         for (int i = 0; i < 2; i++)
287         {
288             if (RS_mul[i].ready_all)//選一個ready進入buffer
289             {
290                 out = true;
291                 RS_posm = i;
292                 buffer_mul.rs = i;
293                 unit_mul = false;
294                 buffer_mul.opert = RS_mul[i].Operator;
295                 buffer_mul.b = RS_mul[i].b;
```

```
295                 buffer_mul.b = RS_mul[i].b;
296                 buffer_mul.c = RS_mul[i].c;
297                 //假設MUL DIV個別需要10/40個cycle
298                 if (buffer_mul.opert == '/')buffer_mul.cycl = 40 - 1;
299                 else buffer_mul.cycl = 10 - 1;
300                 break;
301             }
302         }
303     }
304     else
305     {
306         if (buffer_mul.cycl == 0)//可以wr
307         {
308             out = true;
309             int result;
310             if (buffer_mul.opert == '/')result = buffer_mul.b / buffer_mul.c;
311             else result = buffer_mul.b * buffer_mul.c;
312
313             RS_mul[RS_posm] = {};
314             write_back(buffer_mul.rs + 3, result);//wr to that rs
315             buffer_mul = {};//buffer become empty
316             unit_mul = true;//become empty
317             //checkdis_a = true;
318             dispatch();//unit空間釋出 所以可以找一個ready的作dispatch
319         }
320         else buffer_mul.cycl--;
321     }
322 }
323
```

- Write back:



```
177  void write_back(int rs, int result)//RS=rs+1
178  {
179      //capture
180      for (auto& u : RS_add)//RS中的未知數給值
181      {
182          if (u.ready_all)continue;
183          else if (u.empty)continue;
184
185          if (!u.ready_b && u.b == rs + 1)
186          {
187              u.b = result;
188              u.ready_b = true;//get value ----> become ready
189          }
190          if (!u.ready_c && u.c == rs + 1)
191          {
192              u.c = result;
193              u.ready_c = true;//get value ----> become ready
194          }
195          if (u.ready_b && u.ready_c)
196          {
197              u.ready_all = true;
198              checkdis_a = true;
199          }
200      }
201
202      for (auto& u : RS_mul)//RS中的未知數給值
203      {
204          if (u.ready_all)continue;
205          else if (u.empty)continue;
```



```
206          if (!u.ready_b && u.b == rs + 1)
207          {
208              u.b = result;
209              u.ready_b = true;//get value ----> become ready
210          }
211          if (!u.ready_c && u.c == rs + 1)
212          {
213              u.c = result;
214              u.ready_c = true;//get value ----> become ready
215          }
216          if (u.ready_b && u.ready_c)
217          {
218              u.ready_all = true;
219              checkdis_m = true;
220          }
221      }
222
223      //Update value of RS in RAT
224      bool wr_RF = false;
225      for (int i = 0; i < 5; i++)
226      {
227          if (!RAT[i].empty() && RAT[i][0] == rs + 1)
228          {
229              RAT[i].clear();//釋放RAT
230              wr_RF = true;//update value in RF
231              if (wr_RF)RF[i + 1] = result;
232          }
233      }
234
235  }
```

我會先做 capture，將運算結果給到對應的 RS 中，給值完後更新個別 RS 的狀態，如果 ready 就表示可以 dispatch 了，最後做 RAT 釋放以及 RF 更新。

- 之後呼叫 output 與 cout_RS 輸出結果，並且有一個 bool 值會紀錄是否有變化，有的話輸出，沒有的話則進入下一個 cycle。