

## 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:

Inst.	Is	Ex	Wr	
ADDI F1, F2, 1	1	2	4	SUB ADD: 2
SUB F1, F3, F4	2	4	6	MUL: 10
DIV F1, F2, F3	3	4	44	DIV: 40
MUL F2, F3, F4	4	44	54	
ADD F2, F4, F2	5	55	59	
ADDI F4, F1, 2	6	45	49	
MUL F5, F5, F5	44	54	64	
ADD F1, F4, F4	45	48	50	

※ 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 。

```

8
9  typedef struct A//data
10 {
11     string Reg;
12     int Fa;
13     int Fb;
14     int Fc;
15 }A;
16
17 typedef struct B//rs
18 {
19     char Operator;
20     int b;
21     int c;
22     bool ready_all = false;//true means ready
23     bool ready_b = false;
24     bool ready_c = false;
25     bool empty = true;//true means empty
26 }B;
27
28 typedef struct C//buffer
29 {
30     int rs;
31     int b;
32     char opert;
33     int c;
34     int cycl;//紀錄run幾個cycle後可以wr
35 }C;
36

```

```

37 int RF[6] = { -1,0,2,4,6,8 };//第一個-1不要用
38 vector<int> RAT[5] = {};//始終指每項只能有一個數字
39 //vector<stack<int>>RAT[5];
40 B RS_add[3] = {};
41 B RS_mul[2] = {};
42 int cycle = 0;
43 const int add_cycle = 2, mul_cycle = 10, div_cycle = 40;
44 bool unit_add = true;//ALU unit//true means empty
45 bool unit_mul = true;
46 C buffer_add = {};
47 C buffer_mul = {};
48 bool out = false;//決定output與否
49 int RS_posa, RS_posm;//紀錄現在正在unit執行的RS為RS幾
50 //防止遞迴呼叫時多做
51 bool checkdis_a = false;//true時表示再次呼叫 但總共只讓他做一次
52 bool checkdis_m = false;
53

```

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

```

505     Tomasulo(Data);
506
507     infile.close();
508     return 0;
509 }

```

- Tomasulo:

```

391 void Tomasulo(queue<A>& data)
392 {
393
394     bool first = true; //第一次的話如果可以dispatch要在下一個cycle才可以做
395
396     while (!data.empty())
397     {
398         cycle++;
399
400         if (!first) dispatch();
401         else first = false;
402         A u = data.front();
403         //cout << u.Reg << " " << u.Fa << ", " << u.Fb << ", " << u.Fc << endl;
404         if (issue(u))
405         {
406             data.pop(); //issue successful
407             out = true; //out有變化就要output
408         }
409
410         output();
411         out = false;
412     }
413
414     while (!unit_add || !unit_mul) //unit還有東西所以要繼續算下去
415     {
416         cycle++;
417         dispatch();
418         output();
419         out = false;
420     }
421 }

```

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 capture 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:

```

55 bool issue(A inst)
56 {
57     bool find_empty = false;
58     int whrs; //where is rs
59
60     //RS有空位才可以進
61     if (inst.Reg == "ADDI" || inst.Reg == "ADD" || inst.Reg == "SUB")
62     {
63         for (int i = 0; i < 3; i++) //從第一個開始找有空的RS
64         {
65             if (RS_add[i].empty()) //RSi ----> ex: i=0, 表RS1//要加1
66             {
67                 RS_add[i].empty = false; //更新為不為空
68                 find_empty = true; //RS有位子可以issue
69                 whrs = i + 1;
70
71                 // operator: +
72                 if (inst.Reg == "ADDI" || inst.Reg == "ADD") RS_add[i].Operator = '+';
73                 // operator: -
74                 else RS_add[i].Operator = '-';
75
76                 //check RAT
77                 /*****check inst.b 在RAT中有無東西*****/
78                 if (RAT[inst.Fb - 1].empty()) //RAT沒東西
79                 {
80                     RS_add[i].b = RF[inst.Fb]; //則Fx值為RF裡的值
81                     //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
82                     RS_add[i].ready_b = true; //RS.b is already
83                 }
84                 else //RAT不為空//要等待別人write back
85                 {
86                     //RS_add is not ready
87                     RS_add[i].ready_b = false;
88                     RS_add[i].b = RAT[inst.Fb - 1][0]; //給予RS編號
89                     //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
90                 }
91             }
92         }
93     }
94 }

```

```

88 RS_add[i].b = RAT[inst.Fb - 1][0]; //給予RS編號
89 //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
90 }
91
92 /*****check inst.c 在RAT中有無東西*****/
93 if (inst.Reg == "ADDI")
94 {
95     RS_add[i].c = inst.Fc; //addi , so inst.Fc is a number//ready is already true
96     RS_add[i].ready_c = true;
97 }
98 else //add && sub
99 {
100     if (RAT[inst.Fc - 1].empty()) //RAT沒東西
101     {
102         RS_add[i].c = RF[inst.Fc]; //則Fx值為RF裡的值
103         RS_add[i].ready_c = true;
104     }
105     else //RAT不為空//要等待別人write back
106     {
107         //RS_add is not ready
108         RS_add[i].ready_c = false; //就算RS.b is true但這邊RS.c還是false
109         RS_add[i].c = RAT[inst.Fc - 1][0]; //給予RS編號
110     }
111 }
112
113 /*****inst.a 在RAT中有無RS*****/
114 if (!RAT[inst.Fa - 1].empty()) RAT[inst.Fa - 1].clear(); //若RAT已有RS則先清掉在新增
115 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
116
117 /***更新整體ready sate***/
118 if (RS_add[i].ready_b && RS_add[i].ready_c) RS_add[i].ready_all = true; //already for dispatch
119 break;
120 }
121 }
122

```

```

121 }
122 }
123 else //MUL DIV ----> RS編號+3
124 {
125     for (int i = 0; i < 2; i++) //從第一個開始找有空的RS
126     {
127         if (RS_mul[i].empty())
128         {
129             RS_mul[i].empty = false; //更新為不為空
130             find_empty = true; //RS有位子可以issue
131             whrs = i + 1;
132
133             // operator: +
134             if (inst.Reg == "MUL") RS_mul[i].Operator = '*';
135             // operator: /
136             else RS_mul[i].Operator = '/';
137
138             //check RAT
139             /*****check inst.b 在RAT中有無東西*****/
140             if (RAT[inst.Fb - 1].empty()) //RAT沒東西
141             {
142                 RS_mul[i].b = RF[inst.Fb]; //則Fx值為RF裡的值
143                 RS_mul[i].ready_b = true; //RS.b is already
144             }
145             else //RAT不為空//要等待別人write back
146             {
147                 RS_mul[i].ready_b = false;
148                 RS_mul[i].b = RAT[inst.Fb - 1][0]; //給予RS編號
149             }
150
151             /*****check inst.c 在RAT中有無東西*****/
152             if (RAT[inst.Fc - 1].empty()) //RAT沒東西
153             {
154                 RS_mul[i].c = RF[inst.Fc]; //則Fx值為RF裡的值
155                 RS_mul[i].ready_c = true;
156             }
157             else //RAT不為空//要等待別人write back
158             {
159                 RS_mul[i].ready_c = false;
160                 RS_mul[i].c = RAT[inst.Fc - 1][0]; //給予RS編號
161             }
162
163             //更新整體ready sate
164             if (RS_mul[i].ready_b && RS_mul[i].ready_c) RS_mul[i].ready_all = true; //already for dispatch
165             break;
166         }
167     }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

```

154 RS_mul[i].c = RF[inst.Fc]; //則Fx值為RF裡的值
155 RS_mul[i].ready_c = true;
156 }
157 else //RAT不為空//要等待別人write back
158 {
159     RS_mul[i].ready_c = false; //就算RS.b is true但這邊RS.c還是false
160     RS_mul[i].c = RAT[inst.Fc - 1][0]; //給予RS編號
161 }
162
163 /*****inst.a 在RAT中有無RS*****/
164 if (!RAT[inst.Fa - 1].empty()) RAT[inst.Fa - 1].clear(); //若RAT已有RS則先清掉在新增
165 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
166
167 /****更新整體ready sate****/
168 if (RS_mul[i].ready_b && RS_mul[i].ready_c) RS_mul[i].ready_all = true; //already for dispatch
169 break;
170 }
171 }
172 }
173
174 return find_empty;
175 }

```

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。

```

237 void dispatch()
238 {
239     if (unit_add && checkdis_a) checkdis_a = false;
240     else if (unit_add) //add unit is empty
241     {
242         for (int i = 0; i < 3; i++)
243         {
244             if (RS_add[i].ready_all) //選一個ready進入buffer
245             {
246                 out = true;
247                 RS_posa = i;
248                 buffer_add.rs = i;
249                 unit_add = false;
250                 buffer_add.opert = RS_add[i].Operator;
251                 buffer_add.b = RS_add[i].b;
252                 buffer_add.c = RS_add[i].c;
253                 buffer_add.cycl = 2 - 1; //假設ADD ADDi SUB 均要做2 cycle
254                 break;
255             }
256         }
257     }
258     else
259     {
260         if (buffer_add.cycl == 0) //可以wr
261         {
262             out = true;
263             int result;
264             if (buffer_add.opert == '+') result = buffer_add.b + buffer_add.c; //count the result
265             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
207         if (!u.ready_b && u.b == rs + 1)
208         {
209             u.b = result;
210             u.ready_b = true;//get value ----> become ready
211         }
212         if (!u.ready_c && u.c == rs + 1)
213         {
214             u.c = result;
215             u.ready_c = true;//get value ----> become ready
216         }
217         if (u.ready_b && u.ready_c)
218         {
219             u.ready_all = true;
220             checkdis_m = true;
221         }
222     }
223
224     //Update value of RS in RAT
225     bool wr_RF = false;
226     for (int i = 0; i < 5; i++)
227     {
228         if (!RAT[i].empty() && RAT[i][0] == rs + 1)
229         {
230             RAT[i].clear();//釋放RAT
231             wr_RF = true;//update value in RF
232             if (wr_RF)RF[i + 1] = result;
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。