Notes on solutions to Exercise 4 Graph colouring

Compilers

Paul Kelly

Imperial College London

```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```

```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```

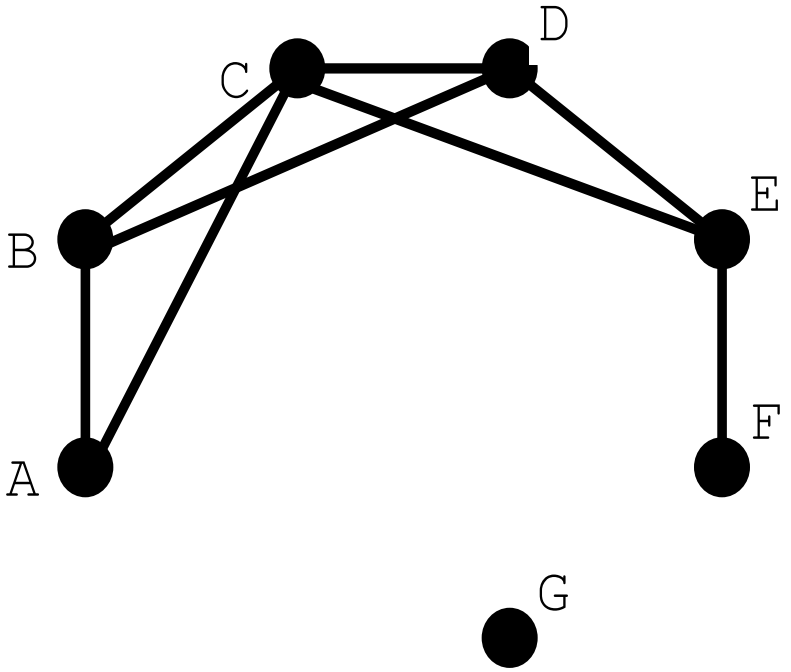
```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```

```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```

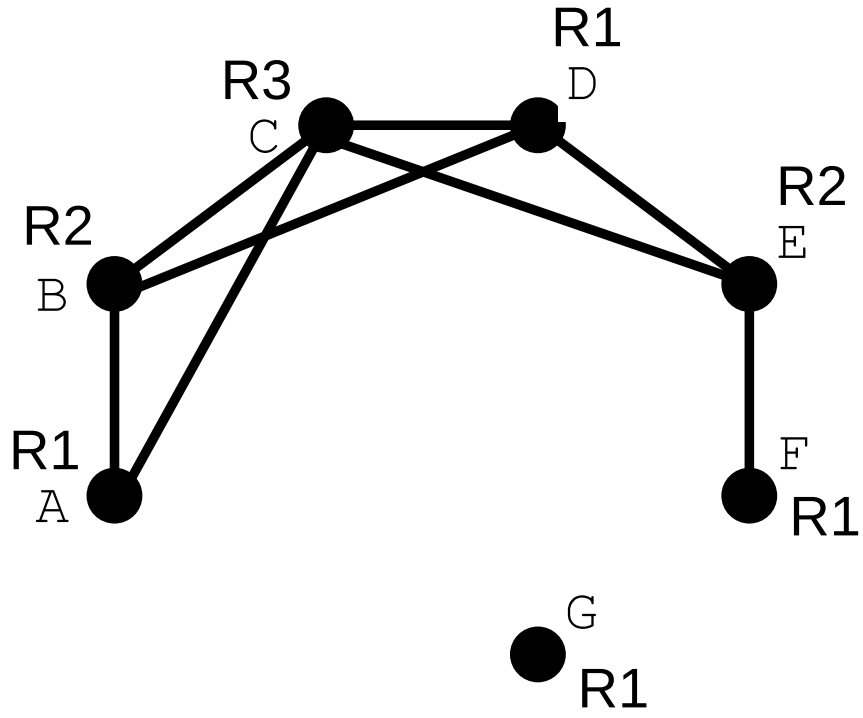
```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```

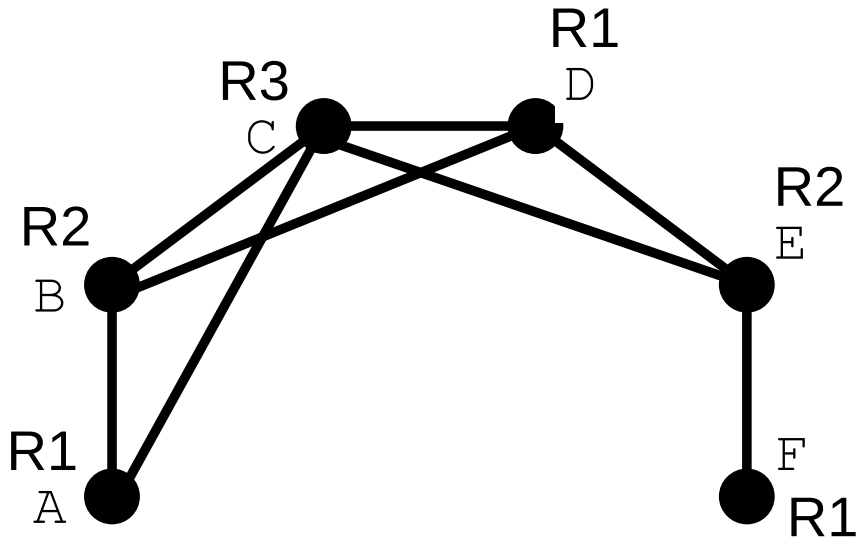
```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```

```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```



```
S1:    A = 100;
P1:
S2:    B = 200;
P2:
S3:    C = A + B;
P3:
S4:    D = A * 2;
P4:
S5:    E = B * 2;
P5:
S6:    F = D - C;
P6:
S7:    G = E + F;
P7:
```





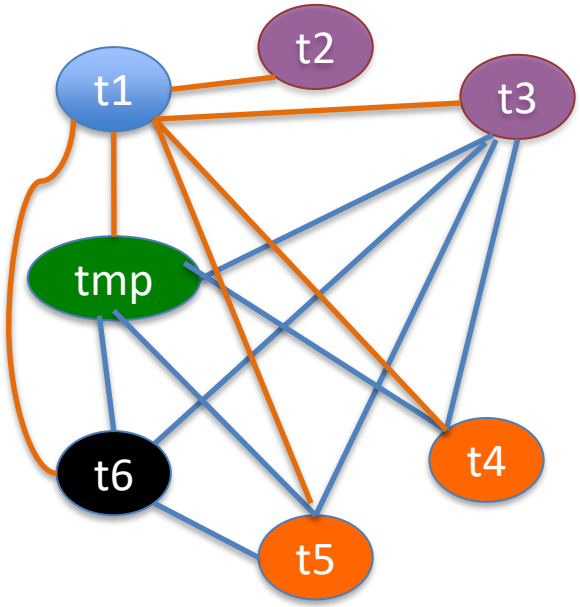
S1:	A = 100	R1 = 100
S2:	B = 200	R2 = 200
S3:	C = A + B	R3 = R1 + R2
S4:	D = A * 2	R1 = R1 * 2
S5:	E = B * 2	R2 = R2 * 2
S6:	F = D - C	R1 = R1 - R3
S7:	G = E + F	R1 = R2 + R1

```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

```
(i coexists with everything
 j coexists with everything
 k coexists with everything)

t1 coexists with everything

t2 coexists with t1
     (and i,j,k but asked to ignore them).
t3 coexists with t1,tmp,t4,t5,t6
tmp coexists with t1,t3,t4,t5,t6
t4 coexists with t1,tmp,t3
t5 coexists with t1,t6,t3,tmp
t6 coexists with t1,t3,tmp,t5
```

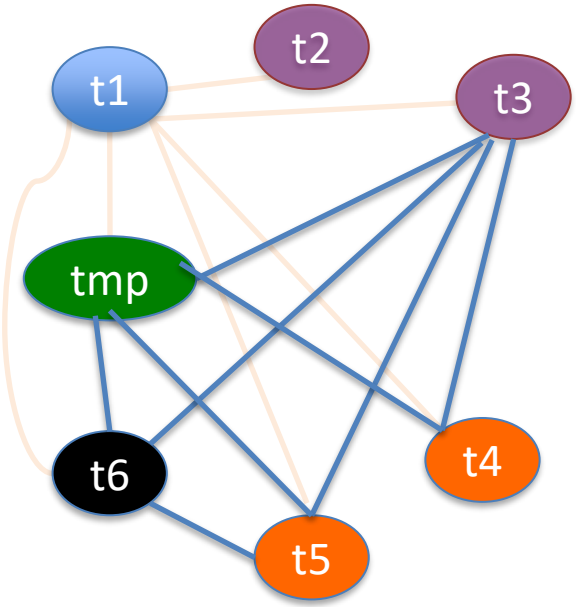


```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

```
(i coexists with everything
 j coexists with everything
 k coexists with everything)

t1 coexists with everything

t2 coexists with t1
     (and i,j,k but asked to ignore them).
t3 coexists with t1,tmp,t4,t5,t6
tmp coexists with t1,t3,t4,t5,t6
t4 coexists with t1,tmp,t3
t5 coexists with t1,t6,t3,tmp
t6 coexists with t1,t3,tmp,t5
```



```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

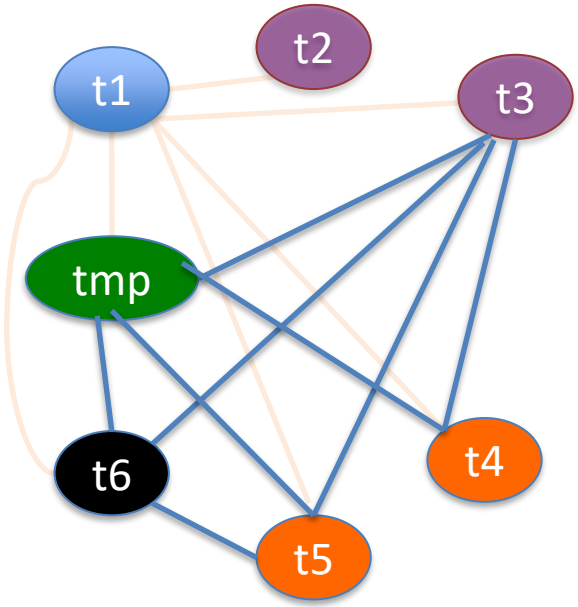
R0: t1

R1: t2, t3

R2: t4, t5

R3: t6

R4: tmp



```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

R0: t1

R1: t2, t3

R2: t4, t5

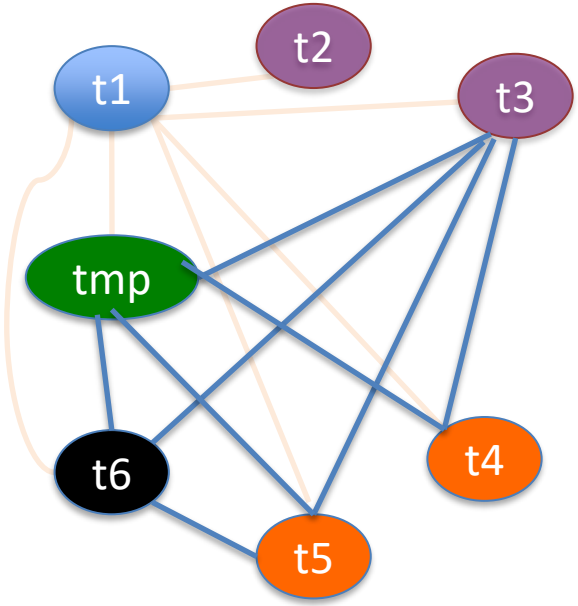
R3: t6

R4: tmp

R5: i

R6: j

R7: k



```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

R0: t1

R1: t2, t3

R2: t4, t5

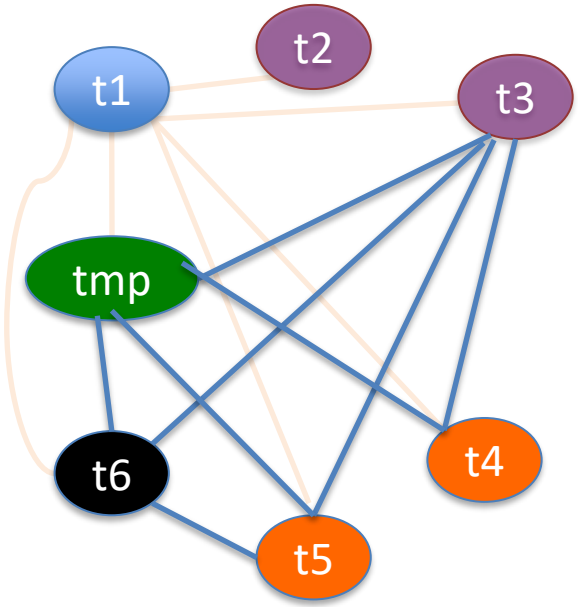
R3: t6

R4: tmp

R5: i

R6: j

R7: k



t2 and t3 can share a register as their live ranges about but do not overlap

```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

R0: t1

R1: t2, t3

R2: t4, t5

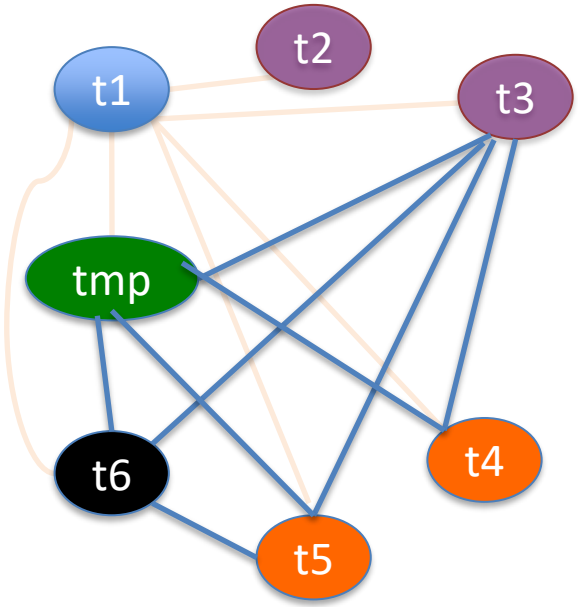
R3: t6

R4: tmp

R5: i

R6: j

R7: k



t4 and t5 can share a register as their live ranges about but do not overlap

```
        R0 := size-1
        R7 := 0
L1:
        cmp R7,R0
        bgt End
        R1 := Address(A)+R5
        R1 := R1+R7
        R4 := LoadIndirect(R1)
        R2 := Address(A)+R6
        R2 := R2+k
        R3 := LoadIndirect(R2)
        StoreIndirect(R3, R1)
        StoreIndirect(R4, R2)
        R7 := R7+1
        jmp L1
End:
```

R0: t1

R1: t2, t3

R2: t4, t5

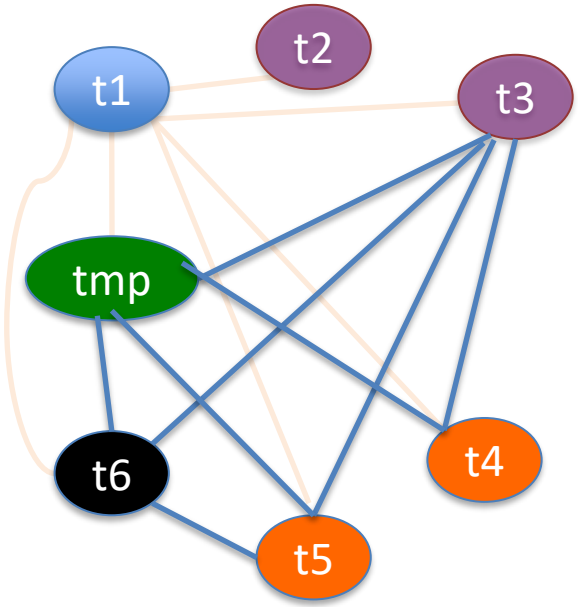
R3: t6

R4: tmp

R5: i

R6: j

R7: k



t2 and t3 can share a register as their live ranges about but do not overlap

```
        t1 := size-1
        k := 0
L1:
        cmp k,t1
        bgt End
        t2 := Address(A)+i
        t3 := t2+k
        tmp := LoadIndirect(t3)
        t4 := Address(A)+j
        t5 := t4+k
        t6 := LoadIndirect(t5)
        StoreIndirect(t6, t3)
        StoreIndirect(tmp, t5)
        k := k+1
        jmp L1
End:
```

R0: t1

R1: t3

R2: t5

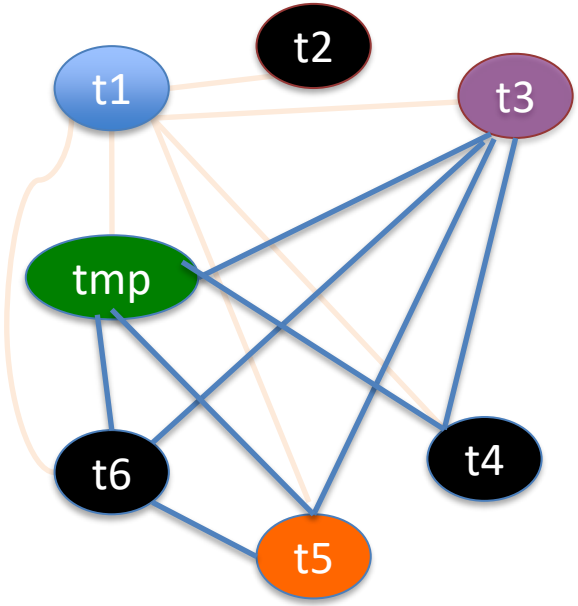
R3: t2, t4, t6

R4: tmp

R5: l

R6: j

R7: k



Other colourings are possible

```
        R0 := size-1
        k := 0
L1:
        cmp R7,R0
        bgt End
        R3 := Address(A)+R5
        R1 := R3+R7
        tmp := LoadIndirect(R1)
        R3 := Address(A)+R6
        R2 := R3+k
        R3 := LoadIndirect(R2)
        StoreIndirect(R3, R1)
        StoreIndirect(tmp, R2)
        R7 := R7+1
        jmp L1
End:
```

R0: t1

R1: t3

R2: t5

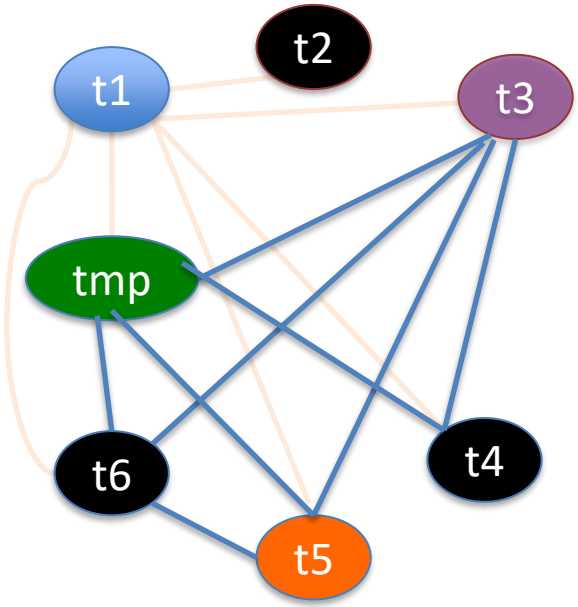
R3: t2, t4, t6

R4: tmp

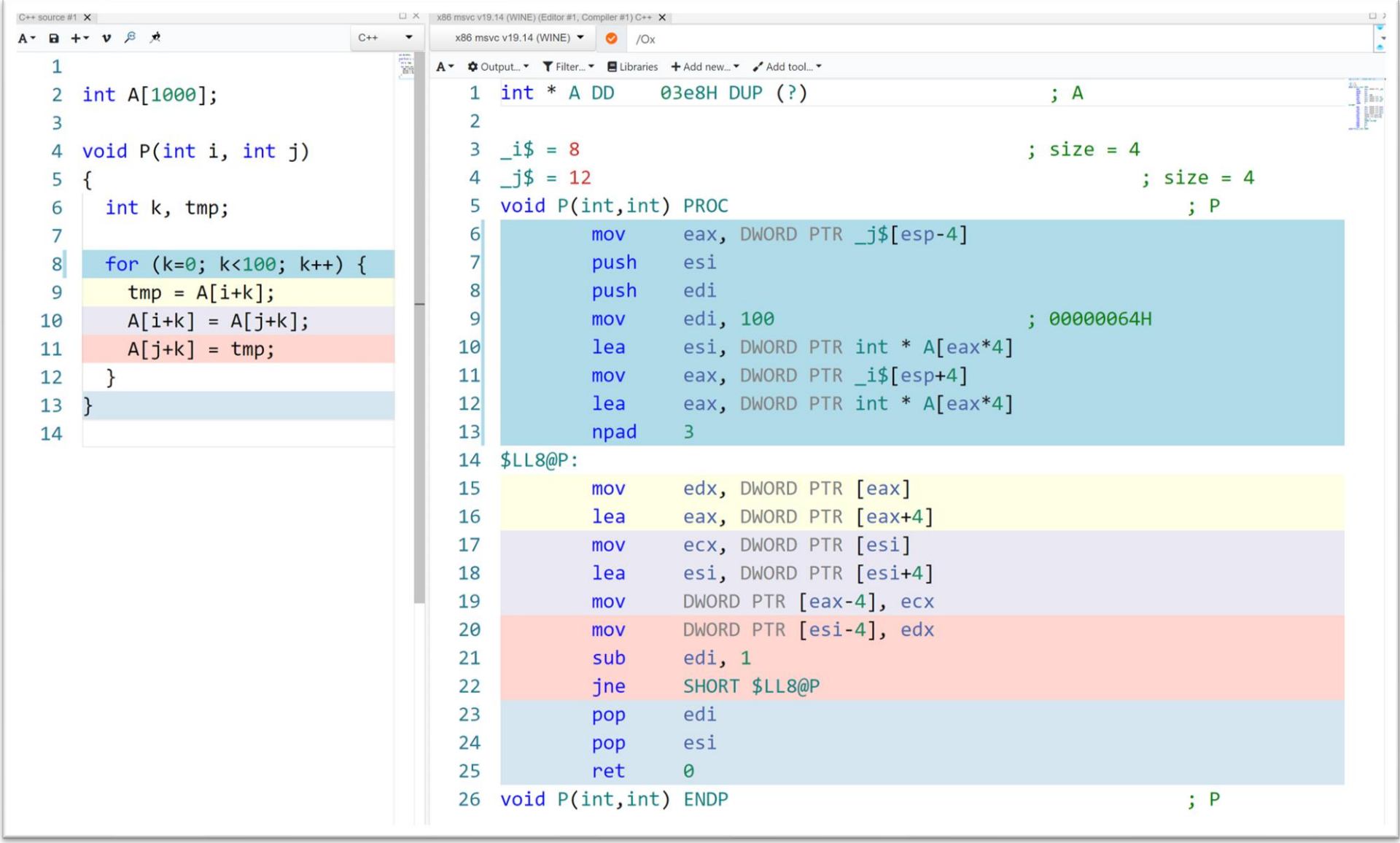
R5: i

R6: j

R7: k



Other colourings are possible

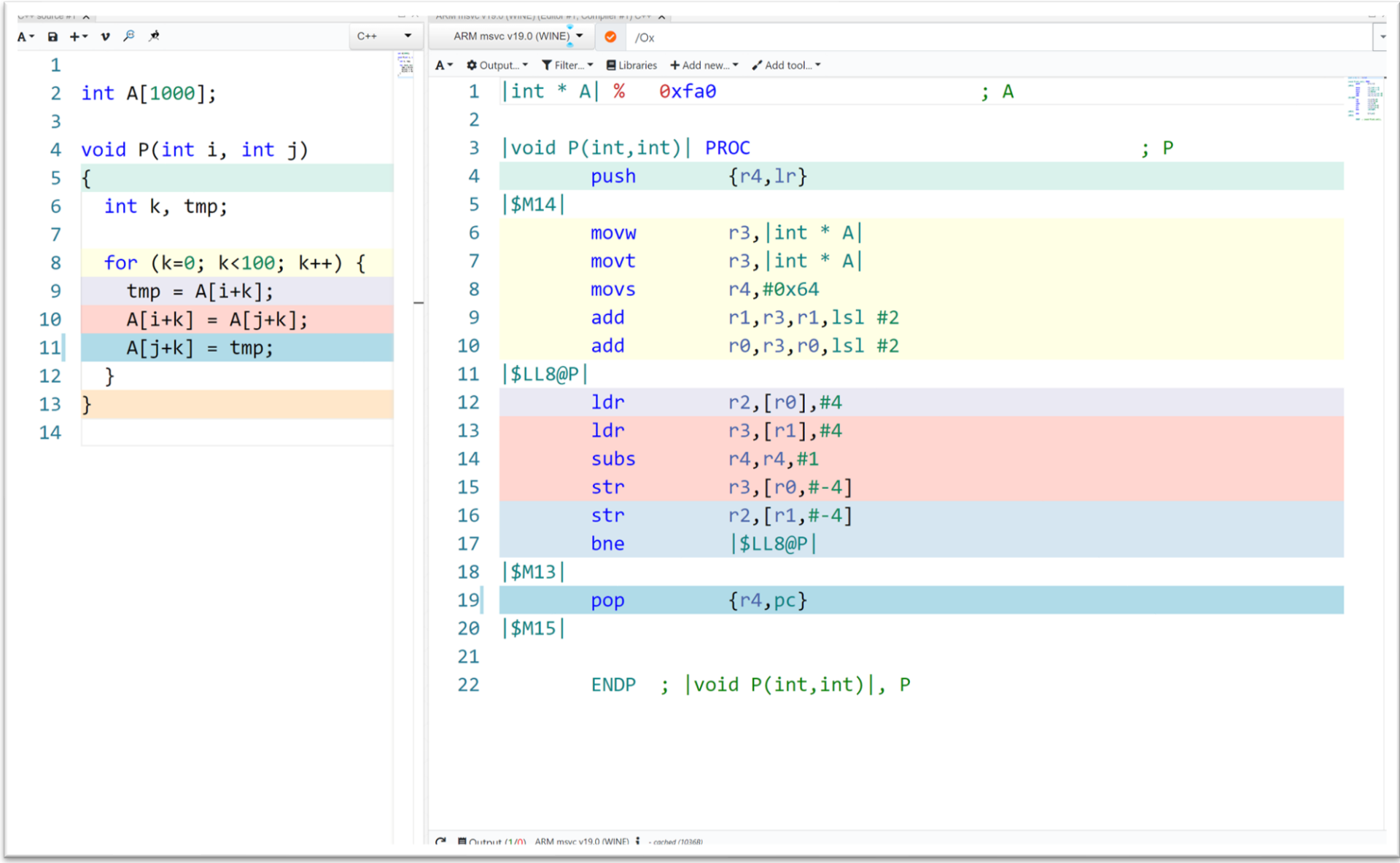


But the number of registers needed can be lower than that!

In this code from the Microsoft MSVC compiler,

5 registers are used: **edx, eax, ecx, esi, edi**

How did they do it?



```
1
2  int A[1000];
3
4  void P(int i, int j)
5  {
6    int k, tmp;
7
8    for (k=0; k<100; k++) {
9      tmp = A[i+k];
10     A[i+k] = A[j+k];
11     A[j+k] = tmp;
12   }
13 }
14
```

```
1  |int * A| %   0xfa0                    ; A
2
3  |void P(int,int)| PROC                          ; P
4          push        {r4,lr}
5  |$M14|
6          movw        r3,|int * A|
7          movt        r3,|int * A|
8          movs        r4,#0x64
9          add         r1,r3,r1,lsl #2
10         add         r0,r3,r0,lsl #2
11 |$LL8@P|
12         ldr         r2,[r0],#4
13         ldr         r3,[r1],#4
14         subs        r4,r4,#1
15         str         r3,[r0,#-4]
16         str         r2,[r1,#-4]
17         bne         |$LL8@P|
18 |$M13|
19         pop         {r4,pc}
20 |$M15|
21
22         ENDP  ; |void P(int,int)|, P
```

But the number of registers needed can be lower than that!

In this code from the Microsoft MSVC compiler for ARM:

5 registers are used: **r0, r1, r2, r3, r4**

How did they do it?