Modulo Scheduling Homework Problem – Solution in Red

Homework Problem

latencies: add=1, mpy=3, ld = 2, st = 1, br = 1

```
for (j=0; j<100; j++)
  b[j] = a[j] * 26
```

```
      LC = 99
Loop: 1: r3 = load(r1)
      2: r4 = r3 * 26
      3: store (r2, r4)
      4: r1 = r1 + 4
      5: r2 = r2 + 4
      7: brlc Loop
```

How many resources of each type are required to achieve an II=1 schedule?
For II=1, each operation needs a dedicated resource, so: 3 ALU, 2 MEM, 1 BR

If the resources are non-pipelined, how many resources of each type are required to achieve II=1
Instead of 1 ALU to do the multiplies, 3 are needed, and instead of 1 MEM to do the loads, 2 are needed.
Hence: 5 ALU, 3 MEM, 1 BR

Assuming pipelined resources, generate the II=1 modulo schedule.
See next few slides

HW continued

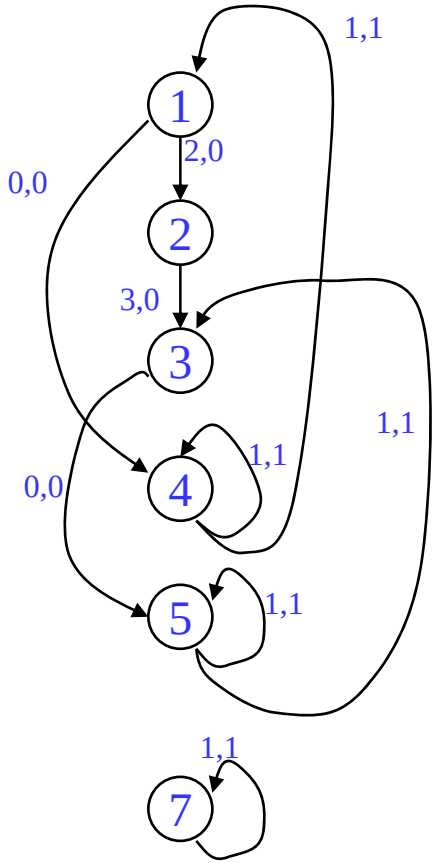
Assume II=1 so resources are: 3 ALU, 2 MEM, 1 BR

DSA converted code below (same as example in class)

LC = 99

```
Loop:  1: r3[-1] = load(r1[0])
       2: r4[-1] = r3[-1] * 26
       3: store (r2[0], r4[-1])
       4: r1[-1] = r1[0] + 4
       5: r2[-1] = r2[0] + 4
       remap r1, r2, r3, r4
       7: brlc Loop
```

Dependence graph (same as example in class)



RecMII = 1
RESMII = 1
MII = MAX(1,1) = 1

Priorities
1: H = 5
2: H = 3
3: H = 0
4: H = 4
5: H = 0
7: H = 0

HW continued

resources: 3 alu, 2 mem, 1 br
latencies: add=1, mpy=3, ld = 2, st = 1, br = 1

LC = 99

Loop:

```
1: r3[-1] = load(r1[0])
2: r4[-1] = r3[-1] * 26
3: store (r2[0], r4[-1])
4: r1[-1] = r1[0] + 4
5: r2[-1] = r2[0] + 4
remap r1, r2, r3, r4
7: brlc Loop
```

Rolled Schedule

0	7 1 4 2 3 5

Unrolled Schedule

0	1 4	stage 1
1		stage 2
2	2	stage 3
3		stage 4
4		stage 5
5	3 5 7	stage 6
6		

Scheduling steps:

Schedule brlc at time II-1
Schedule op1 at time 0
Schedule op4 at time 0
Schedule op2 at time 2
Schedule op3 at time 5
Schedule op5 at time 5
Schedule op7 at time 5

	alu0	alu1	alu2	m1	m2	br
0	X	X	X	X	X	X

MRT

HW continued

The final loop consists of a single MultiOp containing 6 operations, each predicated on the appropriate staging predicate. Note register allocation still needs to be performed.

```
    LC = 99
Loop:
    r3[-1] = load(r1[0]) if p1[0]; r4[-1] = r3[-1] * 26 if p1[2]; store (r2[0], r4[-1]) if p1[5]; r1[-1] = r1[0] + 4 if p1[0]; r2[-1] = r2[0] + 4 if p1[5]; brf Loop
```