Register Renaming Example

This is example program was also used in the handouts on scoreboarding, reservation stations, and register renaming in 1995.

```
1000  LOAD  34(R2),  R6   ;    M[34 + R2] -> B
1001  LOAD  45(R3),  R2   ;    M[45 + R3] -> C
1002  MUL   R2 , R4, R0   ;    C * A      -> D
1003  SUB   R6 , R2, R8   ;    B - C      -> E
1004  DIV   R0 , R6, R10  ;    D / B      -> F
1005  ADD   R8 , R7, R6   ;    B + C      -> G
1006  MUL   R2 , R6, R7   ;    C * B      -> H
```

The example microprocessor has five reservation-stations. Three for the Add/Subtract unit (AddSub1, AddSub2, AddSub3) and two for the Multiply/Divide unit (MultDiv1, MultDiv2).

Given:

- Both loads have issued
- `LOAD` ...`R6` will write to `R10'`
- `LOAD` ...`R2` will write to `R11'`
- `LOAD` ...`R6` completes at time 10
- `LOAD` ...`R2` completes at time 7

Time = 0

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	w	w	w	w	V	w
Restore Value	M	N	O	P	Q	R	S
Phys Reg	R6'	R11'	R9'	R10'	R8'		R7'

PC	1002
PCrestore	801

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	n	n	n	n	n	n	Y	Y	Y	Y	Y	Y
Valid	Y	Y	Y	Y	Y	Y	n	Y	Y	n	n	n
Value								W	X			
PC							801	802	803	804	1000	1001
ArchReg							R0	R10	R7	R4	R6	R2

Time = 1 `T` $\rightarrow$ `R6'` completes
Issue `MUL R2 , R4, R0`

`T` $\rightarrow$ `R6'` completes

1. mark `R6'` as Valid, `R6'` = `T`
2. first three registers on completion list (`R6'`, `R7'`, `R8'`) now valid
3. move values in `R6'`, `R7'`, `R8'` to architected registers `R0`, `R10`, and `R2`
4. `R0`$\rightarrow$`R6'`, `R10`$\rightarrow$`R7'`, and `R2`$\rightarrow$`R8'` (*e.g.* arch regs still waiting for the result of these phys regs), so mark these arch regs as Valid
5. mark `R6'`, `R7'`, `R8'` as idle
6. head of completion list is now `R9'`
7. PCrestore = 804

Issue `MUL R2 , R4, R0`

1. `R2` $\rightarrow$ `R11'` and `R4` $\rightarrow$ `R9'`
2. Head of freelist is `R0'`.
3. Mark destination register (`R0`) as waiting, map R0$\rightarrow$R0".
4. Head of freelist is now `R1'`.
5. So, after renaming we have: `MUL R11', R9', R0'`.
6. Issue `R11'` $*$ `R9'` $\rightarrow$ `R0'` to MulDiv1.

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	w	w	w	V	V	V
Restore Value	T	N	O	P	X	R	W
Phys Reg	R0'	R11'	R9'	R10'			

PC	1003
PCrestore	804

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	Y	n	n	n	n	n	n	n	n	Y	Y	Y
Valid	n	Y	Y	Y	Y	Y	Y	Y	Y	n	n	n
Value												
PC	1002									804	1000	1001
ArchReg	R0									R4	R6	R2

AddSub1	AddSub2	AddSub3	MulDiv1	MulDiv2
			`R11'`$*$`R9'`$\rightarrow$`R0'`	

Time = 2 Issue `SUB R6 R2 R8`

1. `R6`$\rightarrow$`R10'`, `R2`$\rightarrow$`R11'`
2. `R1'` is head of freelist, move head to `R2'`
3. mark `R8` as waiting for `R1'`
4. mark `R1'` as Busy, invalid, arch reg = `R8`
5. issue `R10'`$-$`R11'`$\rightarrow$`R1'` to AddSub1

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	w	w	w	V	w	V
Restore Value	T	N	O	P	X	R	W
Phys Reg	R0'	R11'	R9'	R10'		R1'	

PC	1004
PCrestore	804

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	Y	Y	n	n	n	n	n	n	n	Y	Y	Y
Valid	n	n	Y	Y	Y	Y	Y	Y	Y	n	n	n
Value												
PC	1002	1003								804	1000	1001
ArchReg	R0	R8								R4	R6	R2

AddSub1	AddSub2	AddSub3	MulDiv1	MulDiv2
`R10'`$-$`R11'`$\rightarrow$`R1'`			`R11'`$*$`R9'`$\rightarrow$`R0'`	

Time = 3 Issue `DIV R0 R6 R10`

1. `R0`$\rightarrow$`R0'`, `R6`$\rightarrow$`R10'`
2. `R2'` is head of freelist, move head to `R3'`
3. mark `R10` as waiting for `R2'`
4. mark `R2'` as Busy, invalid, arch reg = `R10`
5. issue `R0'`/`R10'`$\rightarrow$`R2'` to MulDiv2

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	w	w	w	V	w	w
Restore Value	T	N	O	P	X	R	W
Phys Reg	R0'	R11'	R9'	R10'		R1'	R2'

PC	1005
PCrestore	804

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	Y	Y	Y	n	n	n	n	n	n	Y	Y	Y
Valid	n	n	n	Y	Y	Y	Y	Y	Y	n	n	n
Value												
PC	1002	1003	1004							804	1000	1001
ArchReg	R0	R8	R10							R4	R6	R2

AddSub1	AddSub2	AddSub3	MulDiv1	MulDiv2
`R10'`$-$`R11'`$\rightarrow$`R1'`			`R11'`$*$`R9'`$\rightarrow$`R0'`	`R0'`/`R10'`$\rightarrow$`R2'`

Time = 4 `Y`$\rightarrow$`R9'` completes
Issue `ADD R8 R7 R6`

`Y`$\rightarrow$`R9'` completes

1. mark `R9'` as Valid, `R9'` = `Y`
2. move value in `R9'` to `R4`

3. R4→R9′, so mark R4 as Valid
4. mark R9′ as idle
5. head of completion list is now R10′
6. PCrestore = 1000
7. MullDiv1 fetches operand 2

Issue `ADD R8 R7 R6`

1. R8→R1′, R7=X
2. R3′ is head of freelist, move head to R4′
3. mark R6 as waiting for R3′
4. mark R3′ as Busy, invalid, arch reg = R6
5. issue R1′+X→R3′ to AddSub2

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	w	V	w	V	w	w
Restore Value	T	N	Y	P	X	R	W
Phys Reg	R0'	R11'		R3'		R1'	R2'

PC	1006
PCrestore	1000

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	Y	Y	Y	Y	n	n	n	n	n	n	Y	Y
Valid	n	n	n	n	Y	Y	Y	Y	Y	Y	n	n
Value												
PC	1002	1003	1004	1005							1000	1001
ArchReg	R0	R8	R10	R6							R6	R2

AddSub1	AddSub2	AddSub3	MulDiv1	MulDiv2
R10′−R11′→R1′	R1′+X→R3′		R11′*Y→R0′	R0′/R10′→R2′

Time = 5 Attempt to issue `MUL R2 R6 R7`

Can't issue MUL, because both MulDiv reservation stations are occuppied.

Time = 6 Attempt to issue `MUL R2 R6 R7`

Can't issue MUL, because both MulDiv reservation stations are occuppied.

Time = 7 C→R11′ completes
Issue `MUL R2 R6 R7`

C→R11′ completes

1. mark R11′ as Valid, R11′ = C
2. R11′ is *not* head of completion list, so *can not* move value in R11′ to R2
3. AddSub1 fetches operand 2
4. MulDiv1 fetches operand 1 and begins execution
5. MulDiv2 moves to MulDiv1

Issue `MUL R2 R6 R7`

1. R2→R11′ and R11′=C, R6→R3′
2. R4′ is head of freelist, move head to R5′

3. mark `R7` as waiting for `R4'`
4. mark `R4'` as Busy, invalid, arch reg = `R7`
5. issue `C*R3'→R4'` to MulDiv2

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	w	V	w	w	w	w
Restore Value	T	N	Y	P	X	R	W
Phys Reg	R0'	R11'		R3'	R4'	R1'	R2'

PC	1007
PCrestore	1000

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	Y	Y	Y	Y	Y	n	n	n	n	n	Y	Y
Valid	n	n	n	n	n	Y	Y	Y	Y	Y	n	Y
Value												C
PC	1002	1003	1004	1005	1006						1000	1001
ArchReg	R0	R8	R10	R6	R7						R6	R2

AddSub1	AddSub2	AddSub3	MulDiv1	MulDiv2
`R10'−C→R1'`	`R1'+X→R3'`		`R0'/R10'→R2'`	`C*R3'→R4'`

Time = 8, 9 nothing interesting happens

Time = 10 `B→R10'` completes

1. mark `R10'` as Valid, `R10' = B`
2. move values in `R10', R11'` to `R6, R2`
3. `R6→R3'`, not to `R10'` so *do not* mark `R6` as Valid
4. `R2→R11'` so mark `R2` as Valid
5. mark `R10', R11'` as idle
6. head of completion list is now `R0'`
7. PCrestore = 1002
8. AddSub1 fetches operand 1 (`R10'`) and begins execution
9. AddSub2 moves to AddSub1
10. MulDiv1 fetches operand 2 (`R10'`)

Arch Reg	R0	R2	R4	R6	R7	R8	R10
Valid/waiting	w	V	V	w	w	w	w
Restore Value	T	C	Y	B	X	R	W
Phys Reg	R0'			R3'	R4'	R1'	R2'

PC	100?
PCrestore	1002

Phys Reg	R0'	R1'	R2'	R3'	R4'	R5'	R6'	R7'	R8'	R9'	R10'	R11'
Busy	Y	Y	Y	Y	Y	n	n	n	n	n	n	n
Valid	n	n	n	n	n	Y	Y	Y	Y	Y	Y	Y
Value												
PC	1002	1003	1004	1005	1006							
ArchReg	R0	R8	R10	R6	R7							

AddSub1	AddSub2	AddSub3	MulDiv1	MulDiv2
`R1'+X→R3'`			`R0'/B→R2'`	`C*R3'→R4'`
