Arch HW6

1.) a.) C0: R, AC20 → C0.L0: (S, AC20, 20)

b.) C0: W, AC20 → C0.L0: (M, AC20, 80)
↑80 C3.L0: (I, AC20, 20)

c.) C3: W, AC20 → C3.L0: (M, AC20, 80)
↑80

d.) C1: R, AC10 → C1.L2: (S, AC10, 0010)

e.) C0: W, AC08 → C0.L1: (M, AC08, 48)
↑48 C3.L1: (I, AC08, 0008)

f.) C0: W, AC30 → C0.L2: (M, AC30, 78)
↑78 M: AC10 ← 0030

g.) C3: W, AC30 → C3.L2: (M, AC30, 78)
↑78

Arch HW6

2.) a.) P0,0 : read 100, ret 0010

b.) P0,0.B0 : (S, 128, 0028)
L2$,0.B0 : (DS, P0,0, 128, 0028)
M1[128] : (DM, C0, 0028)
P0,0 : read 128, ret 0028

c.) P0,0.B0 : (M, 128, 78)
L2$,0.B0 : (DM, P0,0, 128, 78)
M1[128] : (DM, C0, 78)

d.) P0,0.B0 : (S, 120, 0020)
L2$,0.B0 : (DS, E, 120, 0020)
M1[120] : (DS, C0 C1, 0020)
P0,0 : read 120, ret 0020

e.) P0,0.B0 : (S, 120, 0020)
P1,0.B0 : (S, 120, 0020)
L2$,0.B0 : (DS, E, 120, 0020)
M1[120] : (DS, C0C1, 0020)
P0,0 : read 120, ret 0020
P1,0 : read 120, ret 0020

f.) P0,0.B0 : (I, 120, 0020)
P1,0.B0 : (M, 120, 80)
P3, 1.B0 : (I, 120, 0020)
L2$,0.B0 : (DM, E, 120, 80)
L2$, 1.B0 : (D1, P3,1, 120, 0020)
M1[120] : (DM, C0C1, 80)
P0,0 : read 120, ret 0020

Arch HW 6

2) Continued...

g.) P0,0.B0 : (S, 120, 80)
P1,0.B0 : (S, 120, 80)
P3, 1.B0 : (I, 120, 0020)
L2$, 0.B0 : (DS, E, 120, 80)
L2$, 1.B0 : (DI, P3,1, 120, 0020)
M1[120] : (DS, CO C1, 80)
P0, 0 : read 120, ret 80

h.) P0, 0.B0 : (I, 120, 80)
P1, 0.B0 : (M, 120, 90)
P3, 1.B0 : (I, 120, 0020)
L2$, 0.B0 : (DS, E, 120, 90)
L2$, 1.B0 : (DI, P3,1, 120, 0020)
M1[120] : (DM, CO C1, 90)

3.) a.) P0,0: write 100 ← 80 : Write hit, P0, 0

b.) P0,0 : write 108 ← 88 : Write hit/upgrade, P0,0
invalidate, P3,1

c.) P0,0 : write 118 ← 90 : Write miss, P0,0
invalidate, P1,0

d.) P1,0 : write 128 ← 98 : Write miss, P1,0

Arch HW6

4.) The major change is not needing to access dirty blocks in another CPU's cache. So, in write-through you dont need to provide a forced write-back when reading. When mem is updated with write-through, CPU will always gets the correct data. You dont need to check validity because write-through keeps it always up to date.