v64uint8\_t v64madd(v64uint32\_t acc, v64uint8\_t m1, v1uint8\_t z1, v64uint8\_t m2, v1uint8\_t z2) {

return acc + (m1 – z1) .\* (m2 – z2);

}

v64uint8\_t bc1(v8int64\_t x) {

v64uint8\_t y;

for (int i = 0; i < 8; i++)

y[i][] = x;

return y;

}

v64uint8\_t bc2(v8uint64\_t x) {

v64uint8\_t y;

for (int i = 0; i < 8; i++)

y[][i] = x;

return y;

}

v64uint8\_t bc3(v1uint8\_t x) {

v64uint8\_t y;

for (int i = 0; i < 8; i++)

for (int j = 0; j < 8; j++)

y[i][j] = x;

return y;

}

v64int32\_t bc1(v8int32\_t x) {

v64int32\_t y;

for (int i = 0; i < 8; i++)

y[i][] = x;

return y;

}

v64int32\_t sum\_per\_lane(v64int32\_t x) {

v64int32\_t y;

for (int i = 0; i < 8; i++)

y[i][0] = sum(x[i][]);

return y;

}

v64int32\_t mux\_transpose\_add(v64int32\_t acc1, v64int32\_t acc2, int i1, int i2) {

for (int j = 0; j < 8; j++)

acc1[i1][j] += acc2[j][i2];

return acc1;

}

v64int32\_t qstep(v64int32\_t acc, v64int32\_t A, v1int32\_t B, int i, int j, int shl, int shr) {

return acc + (bytesel(A,i) \* bytesel(B,j)) << (1 + shl)) >> shr;

}

v64int32\_t qstepr(v64int32\_t acc, v64int32\_t A, v1int32\_t B, int i, int j, int shl, int shr) {

return acc + (bytesel(A,i) \* bytesel(B,j)) << (1 + shl)) >>rnd shr;

}

v64uint8\_t acc\_to\_uint8(v64int32\_t x, v1uint8\_t z) {

return int32\_to\_uint8(x – z);  
}

v64uint8\_t select(v8bool sel, v64uint8\_t x, v64uint8\_t y) {

v64uint8\_t z;

for (int i = 0; i < 8; i++)

z[i][] = sel[i] ? x[i][] : y[i][];

return z;

}

chess\_vector\_ptr<T> = T@

void chess\_storage(VM)\* extv1(void chess\_storage(VM)@ v8addr, int i) {

return v8addr[i];

}

void chess\_storage(VM)@ updv1(void chess\_storage(VM)@ v8addr, int i, void chess\_storage(VM)\* addr) {

v8addr[i] = addr;

return v8addr;

}

void masked\_store(v8bool sel, v8uint8\_t chess\_storage(VM)@ v8addr, v64uint8\_t x) {

for (int i = 0; i < 8; i++)

if (sel[i])

VMv[v8addr[i]] = x[i][];

}

v64uint8\_t masked\_load(v8bool sel, v8uint8\_t chess\_storage(VM)@ v8addr, v64uint8\_t x) {

v64uint8\_t y;

for (int i = 0; i < 8; i++)

y[i][] = sel[i] ? VMv[v8addr[i]] : x[i][];

return y;

}