Why does attention have an output projection?

Heads are independent and additive: what W_O really does. Based on the framing in *A Mathematical Framework for Transformer Circuits* (Elhage et al., 2021); the toy numbers below were checked with numpy.

1. Notation (read this first)

Convention: **one row per token**. Matrices multiply on the right: a token's vector is a row, and `X · W` applies W to every row.

Symbol	Meaning	Shape
`n`	number of tokens in the sequence	–
`d`	model width (the size of the residual stream)	–
`H`, `k`	number of heads; width of one head (`k = d / H` in most models, so `H·k = d`)	–
`X`	layer input: one row per token (the residual stream at this layer)	n × d
Wq_h, Wk_h, Wv_h	head h's query, key and value weights	d × k
P_h	head h's attention pattern: softmax($X{\cdot}Wq_h \cdot (X{\cdot}Wk_h)^T$ / √k + causal mask). Row i = how token i spreads its attention over earlier tokens. Depends on head h's Q and K only.	n × n
Z_h	head h's output: $P_h \cdot X \cdot Wv_h$ (a weighted average of value vectors)	n × k
$[Z_1 \mid Z_2 \mid \ldots]$	concatenation: place the Z's side by side (columns)	n × (H·k)
W_0	the output projection, one matrix for the whole layer	(H·k) × d
$W_{0,h}$	rows $(h-1){\cdot}k+1 \ldots h{\cdot}k$ of W_O: head h's own slice; stacking $W_{O,1}$ on top of $W_{O,2}$ … gives W_O	k × d
OV_h	$Wv_h \cdot W_{0,h}$: what head h does to a token vector, end to end	d × d, rank ≤ k

2. The layer, standard form and the equivalent form

> **Standard form (how it is implemented, one big matmul):**
> $\mathrm{Out} = [Z_1 \mid Z_2 \mid \ldots \mid Z_H] \cdot W_O$ (n × d), then added to the residual stream: $X_{next} = X + \mathrm{Out}$.

> **Equivalent form (the key identity):**
> $\mathrm{Out} = Z_1 \cdot W_{O,1} + Z_2 \cdot W_{O,2} + \ldots + Z_H \cdot W_{O,H} = \Sigma_h\, P_h \cdot X \cdot OV_h$

Why it holds (2 heads): splitting a product by blocks. A row $[a\ b]$ times a stacked matrix $[U;\ V]$ equals $a \cdot U + b \cdot V$, because each output entry is a sum over the inputs' positions, and the sum over the first k positions and the sum over the last k positions just add. So $[Z_1 | Z_2] \cdot [W_{O,1};\ W_{O,2}] = Z_1 W_{O,1} + Z_2 W_{O,2}$. The second equality uses $Z_h\ W_{O,h} = P_h\ X\ (Wv_h\ W_{O,h})$.

3. Toy numbers (d = 4, H = 2, k = 2, n = 2 tokens)

Head 1 is blue (columns 1-2 of the concatenation, rows 1-2 of W_O), head 2 is orange. Patterns are given rather than computed: P_1 lets token 2 average tokens 1 and 2; P_2 lets each token look only at itself.

$$X = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \qquad P1 = \begin{bmatrix} 1 & 0 \\ 0.5 & 0.5 \end{bmatrix} \qquad P2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Wv1\ (4\times2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \qquad Wv2\ (4\times2) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Head outputs $Z_h = P_h \cdot X \cdot Wv_h$ (each n × k):

$$Z1 = \begin{bmatrix} 3 & 2 \\ 2 & 2 \end{bmatrix} \qquad Z2 = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$$

Way A, concatenate then project: $[Z_1|Z_2] \cdot W_0$

$$\underbrace{\begin{bmatrix} 3 & 2 & 1 & 2 \\ 2 & 2 & 3 & 2 \end{bmatrix}}_{[Z1 \mid Z2]} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} = \underbrace{\begin{bmatrix} 4 & 3 & 5 & 0 \\ 5 & 5 & 4 & 0 \end{bmatrix}}_{\text{Out}}$$

W_O (4x4): top 2 rows = W_O,1, bottom 2 rows = W_O,2

Way B, each head writes its own contribution, then add: $Z_1W_{0,1} + Z_2W_{0,2}$

$$\underbrace{\begin{bmatrix} 3 & 2 & 3 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix}}_{Z1 \cdot W_O,1} + \underbrace{\begin{bmatrix} 1 & 1 & 2 & -2 \\ 3 & 3 & 2 & -2 \end{bmatrix}}_{Z2 \cdot W_O,2} = \underbrace{\begin{bmatrix} 4 & 3 & 5 & 0 \\ 5 & 5 & 4 & 0 \end{bmatrix}}_{\text{Out (same as A)}}$$

Way C, with the single OV matrices: $Wv_h \cdot W_{0,h}$ (4 × 4, but rank 2 each):

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{OV1} \quad \underbrace{\begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 \end{bmatrix}}_{OV2}$$

and $P_1 \cdot X \cdot OV_1 + P_2 \cdot X \cdot OV_2$ also gives Out exactly (verified).

4. What this means

- **Each head works on its own inside a layer, and W_O combines them.** Head h's pattern uses only its own Q and K, and its output goes into the sum linearly. Read through W_O, the sum is also a mixing of the heads' outputs: every output coordinate is a learned combination of all the heads. The heads meet in the residual stream, where a later head reads what earlier heads wrote (the paper calls this Q-, K- and V-composition).
- **W_O decides where each head writes.** Without W_O, the concatenation puts head 1 in coordinates 1-2 and head 2 in coordinates 3-4, always. In the toy: head 1's write would be Z_1 in columns 1-2 only. With $W_{O,1}$ it is `[3 2 3 2; 2 2 2 2]`: a different 2-dimensional slice of the 4-dimensional residual stream, free to overlap with other heads' slices and with what the MLPs and embeddings write.
- **Only the product $OV_h = Wv_h \cdot W_{O,h}$ matters.** You can replace Wv_h by $Wv_h \cdot R$ and $W_{O,h}$ by $R^{-1} \cdot W_{O,h}$ for any invertible k × k matrix R and the model is identical (verified in the toy with R = [[1,1],[0,1]]). So "V" and "O" are one linear map split in two for dimension bookkeeping: d → k, then k → d.

- **Same split on the other side:** the QK circuit is $W_{q_h} \cdot W_{k_h}^T$ (d × d, rank ≤ k). It decides *where* head h looks; the OV circuit decides *what* it moves.

5. Your two ideas from the thread, scored

Idea	Verdict
"With separate queries per head there is less cross-mixing, so W_O compensates."	W_O does mix the heads' outputs into one vector per token, and this is the same thing as each head adding its own write. Cross-head effects on the attention patterns arise in later layers, which read the residual stream.
"It is like the last layer of an MLP: heads are hidden neurons, W_O is the down-projection."	Half right. Both are parallel units summed into one shared space through a learned output matrix. But a head writes a rank-k chunk whose content depends on the attention pattern, not a scalar times a fixed vector.

6. Caveats

- I did not find (and am not citing) a clean experiment measuring how much quality is lost if W_O is dropped. In pure linear algebra a later reader could absorb a change of basis, so W_O is a *freedom to choose the write subspace*, not a mathematical necessity. The shared basis matters in practice because MLP activations and normalisation are not basis-independent.
- The paper analyses attention-only transformers and mostly ignores MLPs and LayerNorm. It does not cover RoPE, GQA, MLA or KV-cache cost.
- In llama.cpp, W_O is the single `attn_output` matmul on the concatenated head outputs; the per-head split above is for understanding only.

Suggested reading: the paper's sections on the residual stream and on "attention heads are independent and additive", and the QK/OV circuit decomposition. All toy numbers were checked in numpy: Way A, B and C agree exactly, and the R-gauge check passes.