Clipping

- **Any procedure that identifies those portions of a picture that are either inside or outside of a specified region**

- **Point clipping**
- **Line clipping**
- **Area clipping or polygon clipping**
- **Curve clipping**
- **Text clipping**

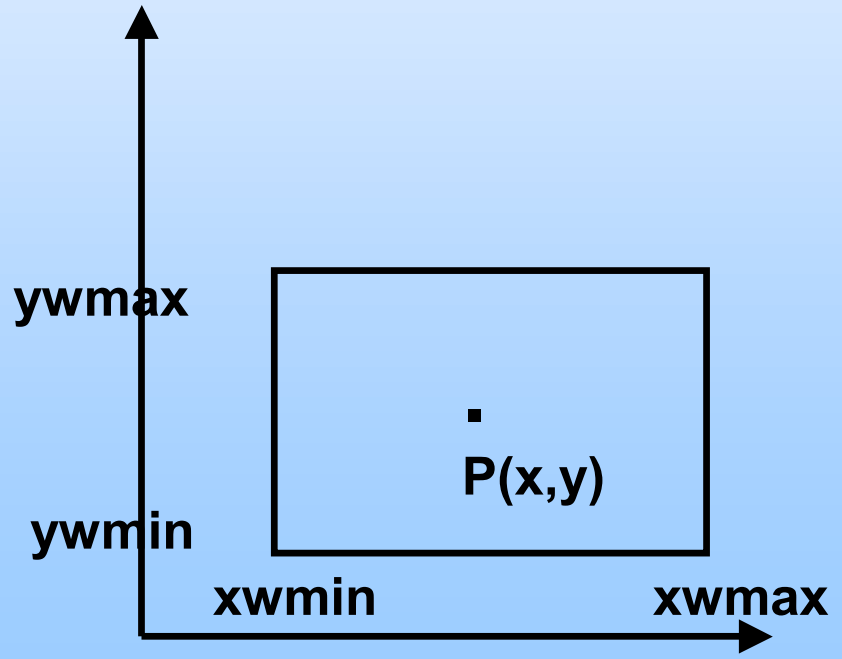
Point clipping

- **In a rectanguar clip window save a point**

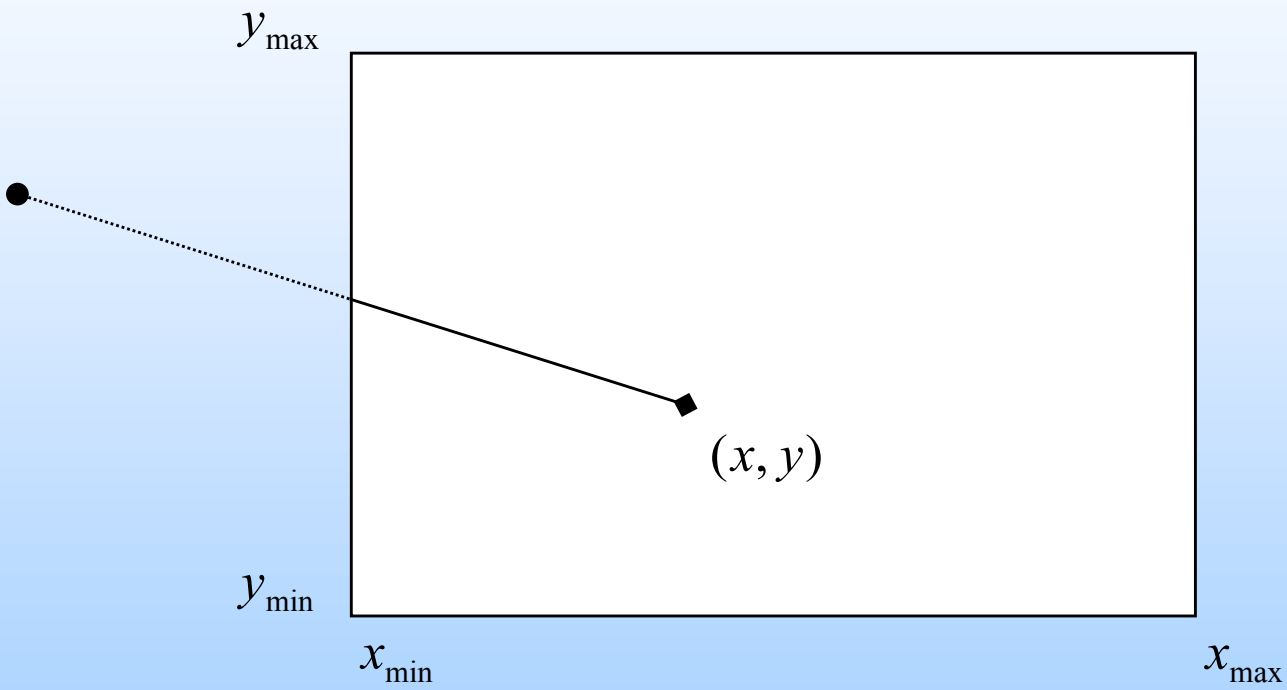
P = (x, y) for display if

$$Xw_{min} \leq x \leq xw_{max}$$

$$yw_{min} \leq y \leq yw_{max}$$



Inside/Outside Test



$$x_{min} \leq x \leq x_{max}$$

$$y_{min} \leq y \leq y_{max}$$

Line Clipping

- Check whether the line is completely outside or inside the window boundary.
- Then perform inside outside test.
- Parametric eqn of a line with endpoints (x1,y1) and (x2,y2) is

$x = x1 + u (x2-x1)$

Where $u=(y – y1)/(y2 – y1)$ (horozontal intersection)

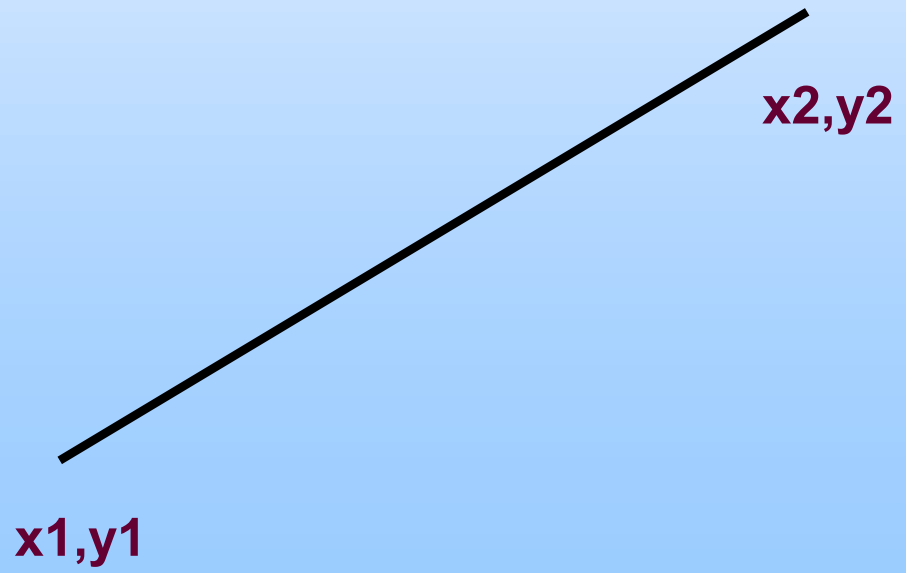
And $y=ymin$ or $y=ymax$

$y = y1 + u (y2-y1)$

$u=(x-x1)/(x2-x1)$ (vertical intersection)

And $x=xmin$ or $x=xmax$

Where $0<u<1$



Clipping Algorithms

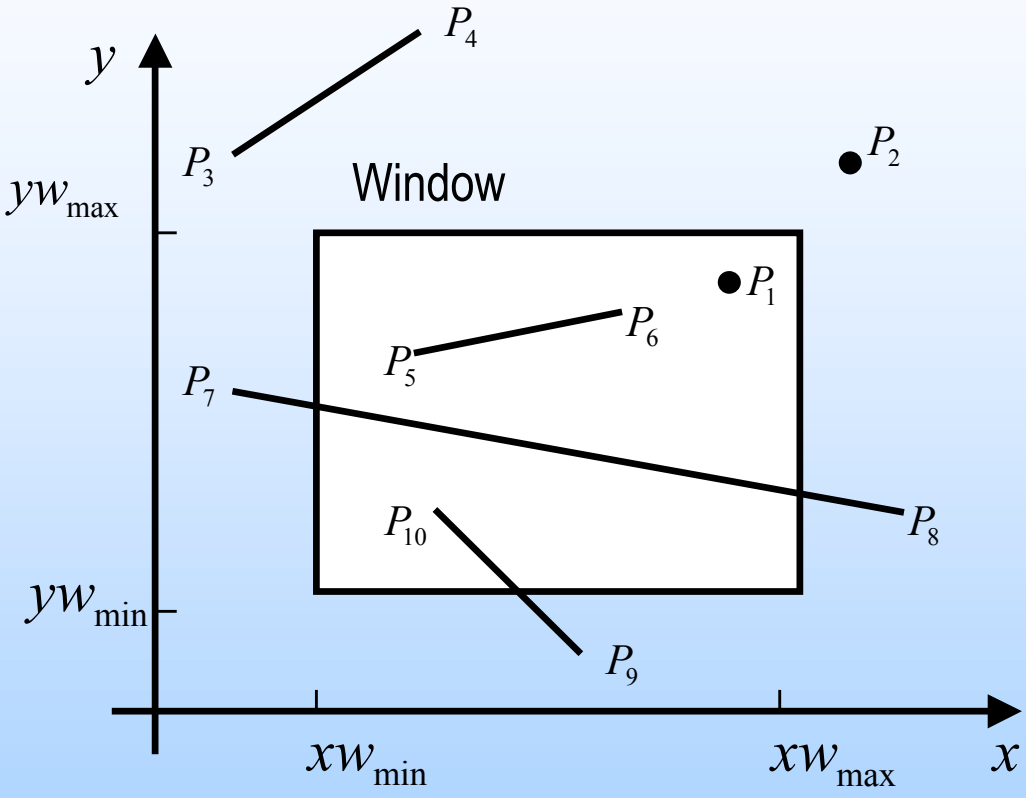
Line Clipping:

- Cohen-Suterland (encoding) -Oldest and most commonly used
- Nicholl-Lee-Nicholl (encoding) (more efficient)
- Liang-Barsky (parametric)- More efficient than Cohen-Sutherland

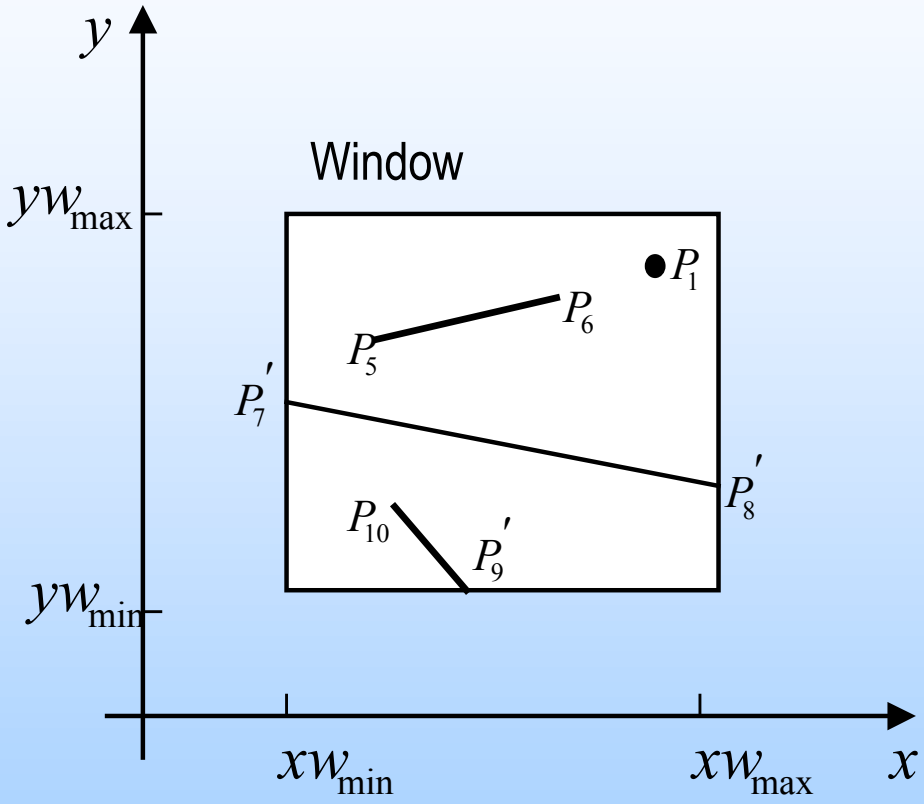
Polygon Clipping:

- Sutherland-Hodgeman (divide and conquer strategy)
- Weiler-Atherton (modified for concave polygons)

Line Clipping



(a) Before Clipping



(b) After Clipping

$$x_{min} \le x \le x_{max}$$

$$y_{min} \le y \le y_{max}$$

Sutherland and Cohen 2D Clipping Algorithm

Basic Idea

- **Encode the line endpoints**
- **Successively divide the line segments so that they are completely contained in the window or completely lies outside window**
- **Division occurs at the boundary of window**

Region Code Setting

- Calculate the difference between endpoint coordinates and clipping boundaries.

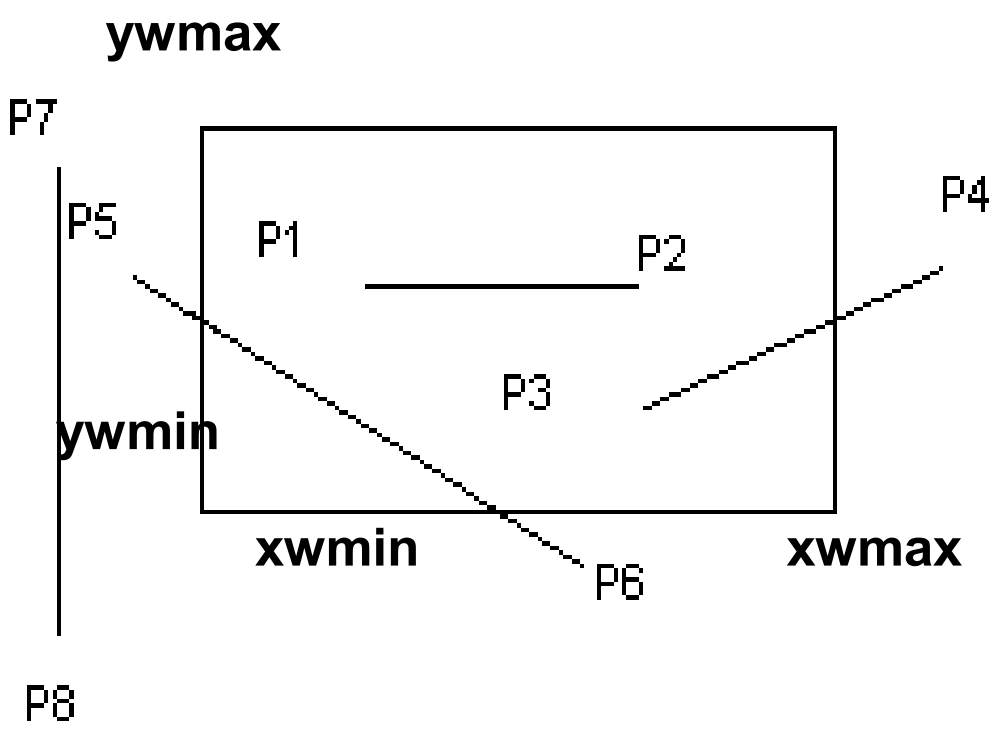
- Use the resultant sign bit of each difference calculation to set corresponding region.

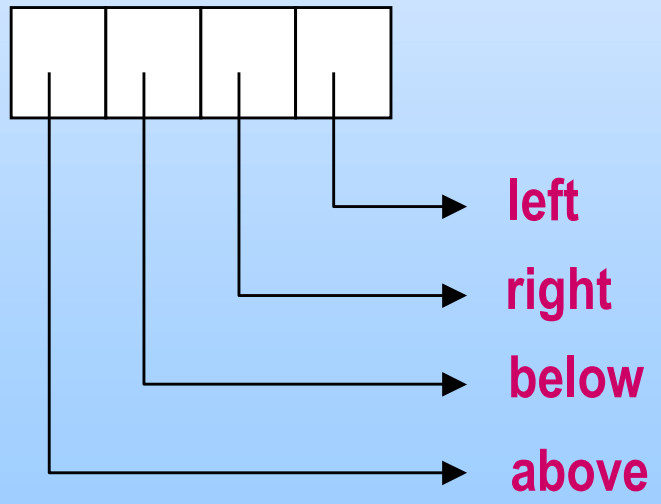
- Bit 1 sign bit of $x - xw_{min}$
- Bit 2 sign bit of $xw_{max} - x$
- Bit 3 sign bit of $y - yw_{min}$
- Bit 4 sign bit of $yw_{max} - y$

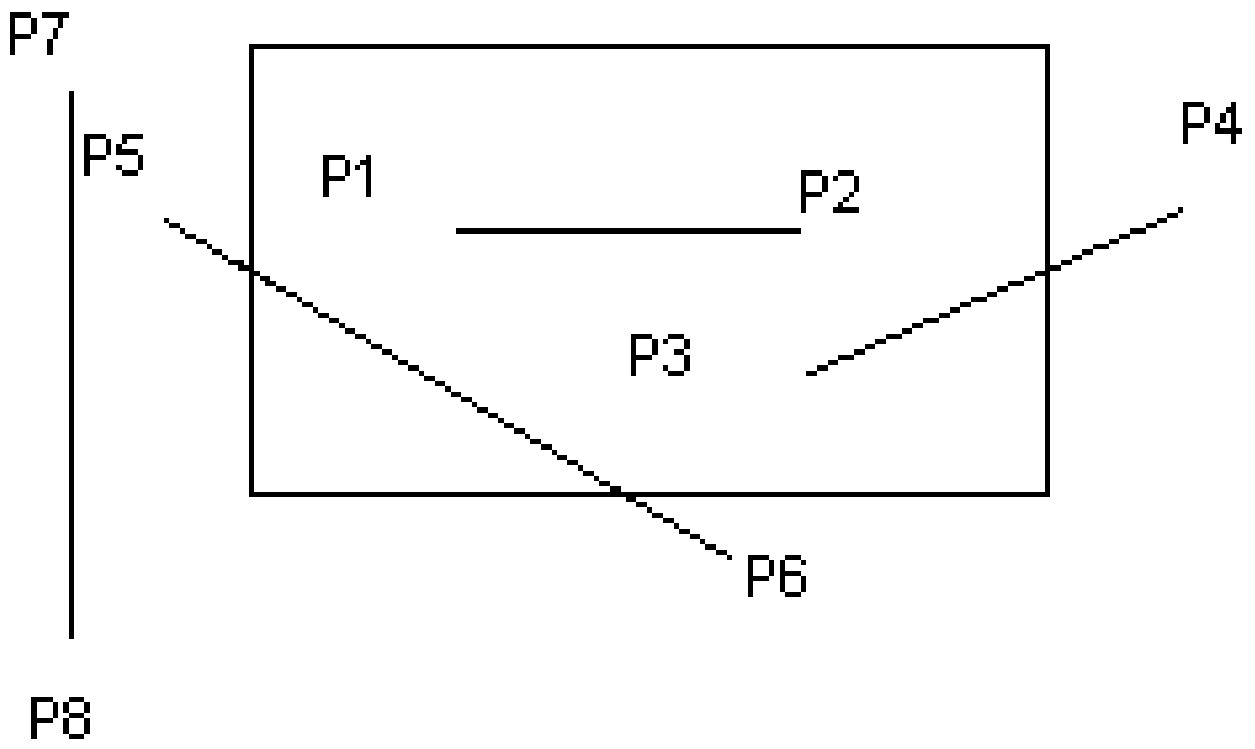
OR

- if(x<Xwmin) then Bit 1
- if(x>Xwmax) then Bit 2
- if(y<Ywmin) then Bit 3
- if(y>Ywmax) then Bit 4

1001	1000	1010
0001	0000	0010
0101	0100	0110







Bit 1- left

Bit 2- right

Bit 3- below

Bit 4 - Above

P1,P2 - 0000

P3 - 0000

P4 - 0010

P5 - 0001

P6 - 0100

P7 - 0001

P8 - 0101

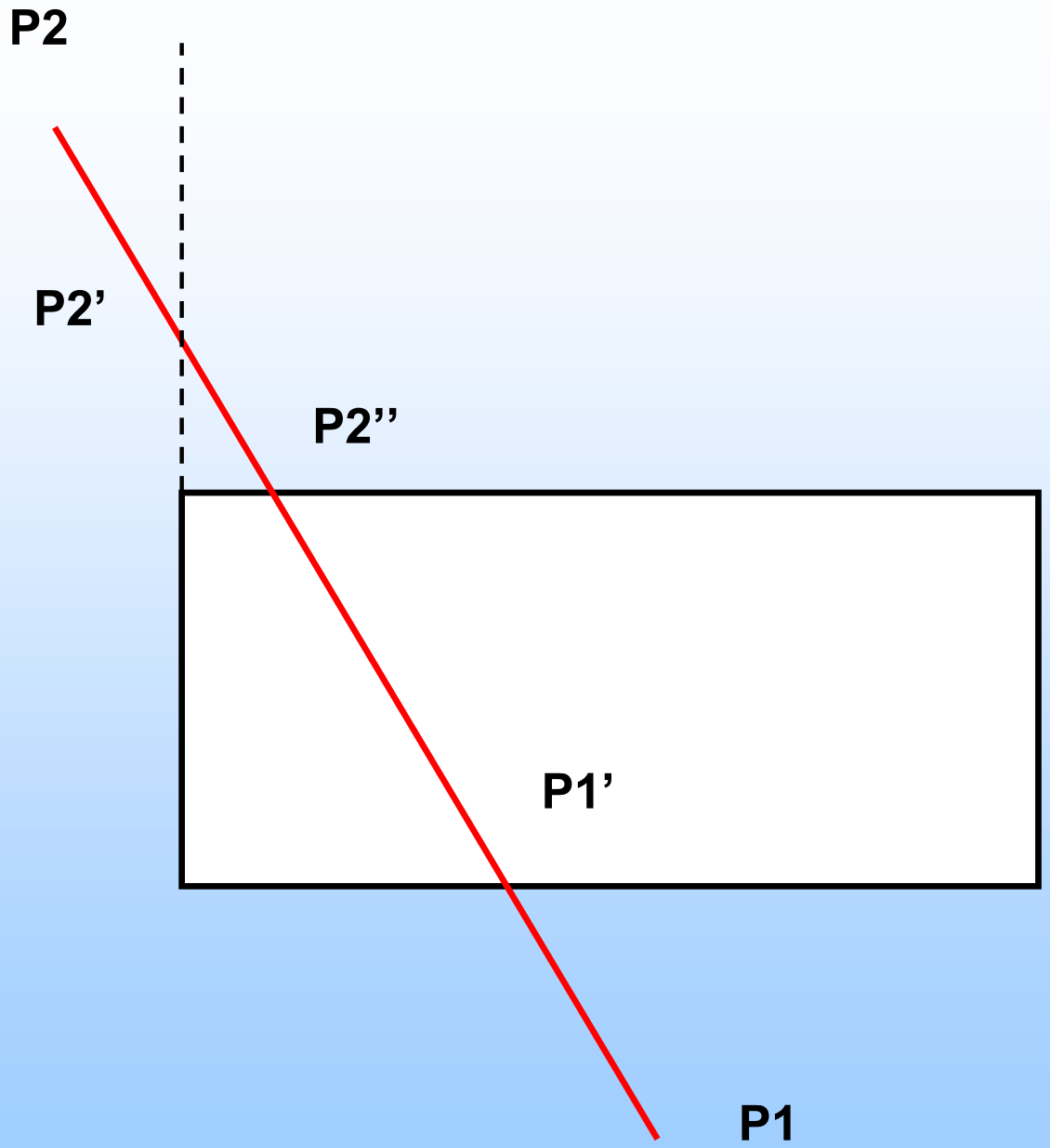
- **After setting the region code determine whether the lines are**
  - **Completely inside the clip window.**
  - **Completely outside the clip window.**
  - **Intersecting with the window boundaries.**

- Completely Inside
  - Region code 0000 for both the endpoints.
  - Accept the line

- Completely outside
  - Lines have 1 in the same bit position.
  - Reject the line.

Method to test for total clipping

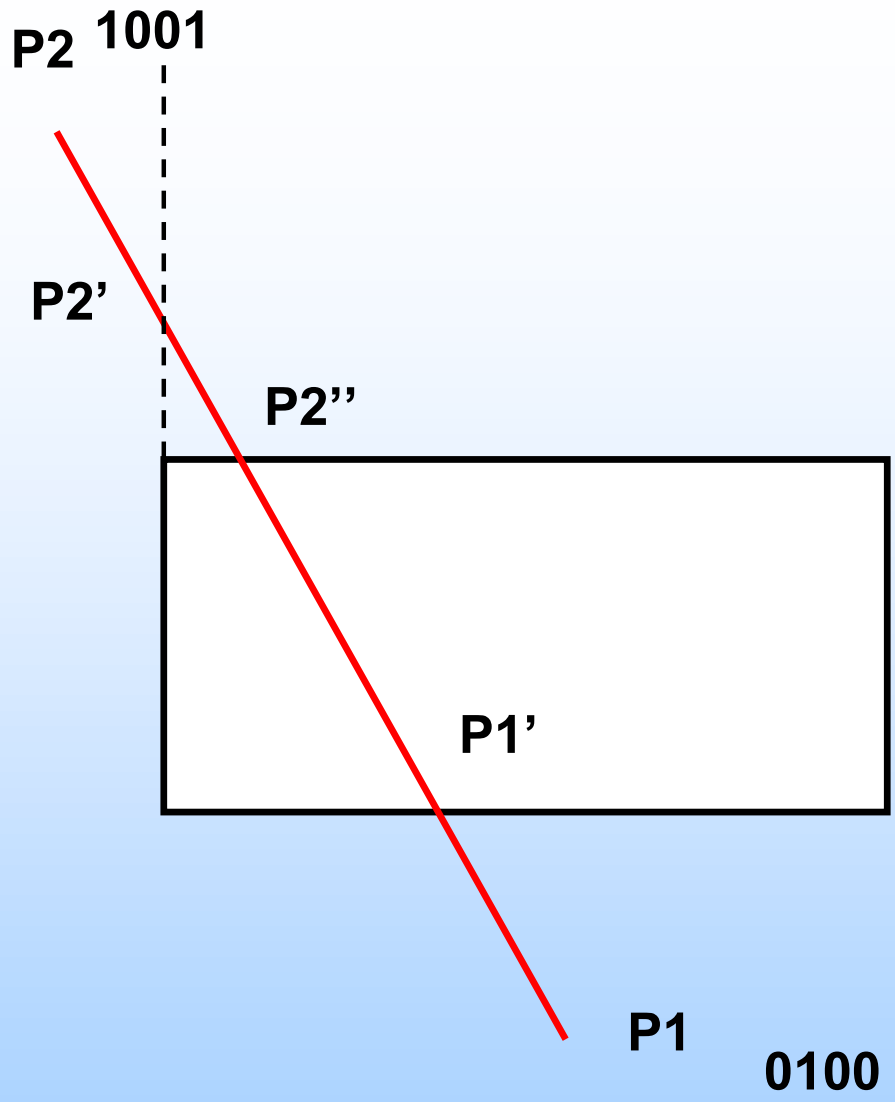
- Perform logical AND operation with the region code.
- If the result is not 0000 line is completely outside the clipping region.
- Lines that are not completely outside or inside a clip window are checked for intersection with the window boundary.


P2
P2’
P2’’
P1’
P1

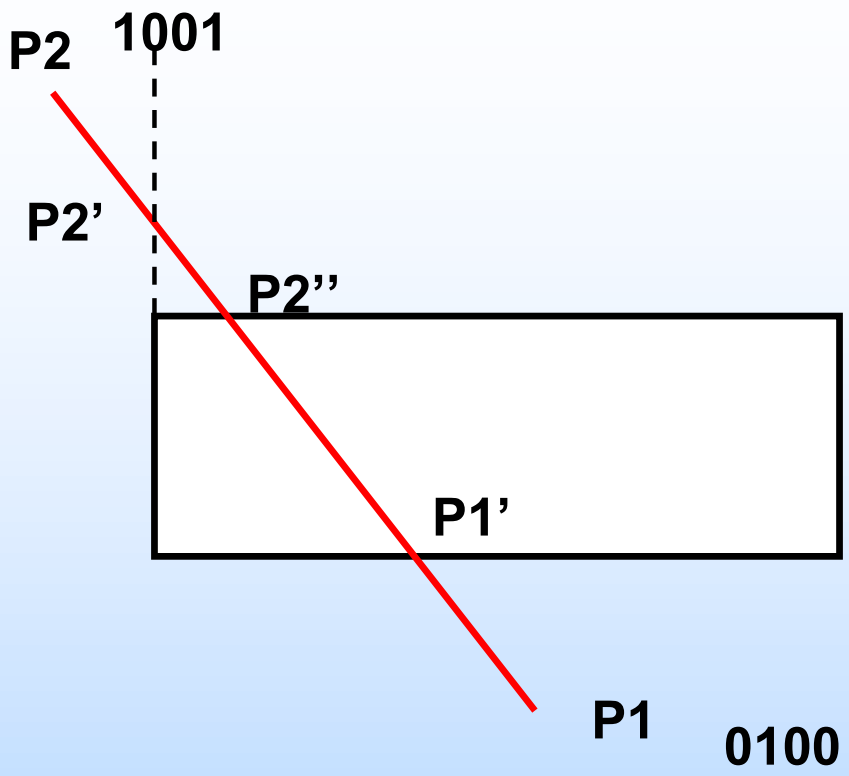
Steps

- Lines may or may not cross the window interior.
- Start from an outside end point and check with the clipping boundary to determine how much of the line can be discarded.
- The remaining part of the line is checked against other boundaries & continue until either the line is totally discard or a section is found inside the window.
- Check line end points against clipping boundaries in the order
  - Left, Right, Bottom, top

- Check P1 against left, right & bottom boundary.
- Find intersection point P1’ with bottom boundary and discard line section from P1 to P1’.
- Line reduced to P1’ P2.



- Check P2 against the boundaries.
- P2 to the left of the window. 1001
- Intersection point P2’ is calculated.
- P2’ above the window.
- Final intersection calculation leads to p2’’.
- Line from P1’ to P2’’ s saved



- Intersection points with a boundary is calculated using slope intercept form

- Intersections with a vertical boundary:
  $y=y1+m(x-x1)$ (x is either xmin or xmax)

- Intersections with a horizontal boundary:
  $x=x1+(y-y1)/m$ (y is either ymin or ymax)

Continue until a trivial accept or a trivial reject.

Cohen-Sutherland Line-Clipping

- Will do unnecessary clipping.
- Not the most efficient.
- Clipping and testing are done in fixed order.
- Efficient when most of the lines to be clipped are either rejected or accepted (not so many subdivisions).
- Easy to program.
- Parametric clipping are more efficient.