GREP in Assembler

Steven R. Bagley

Introduction

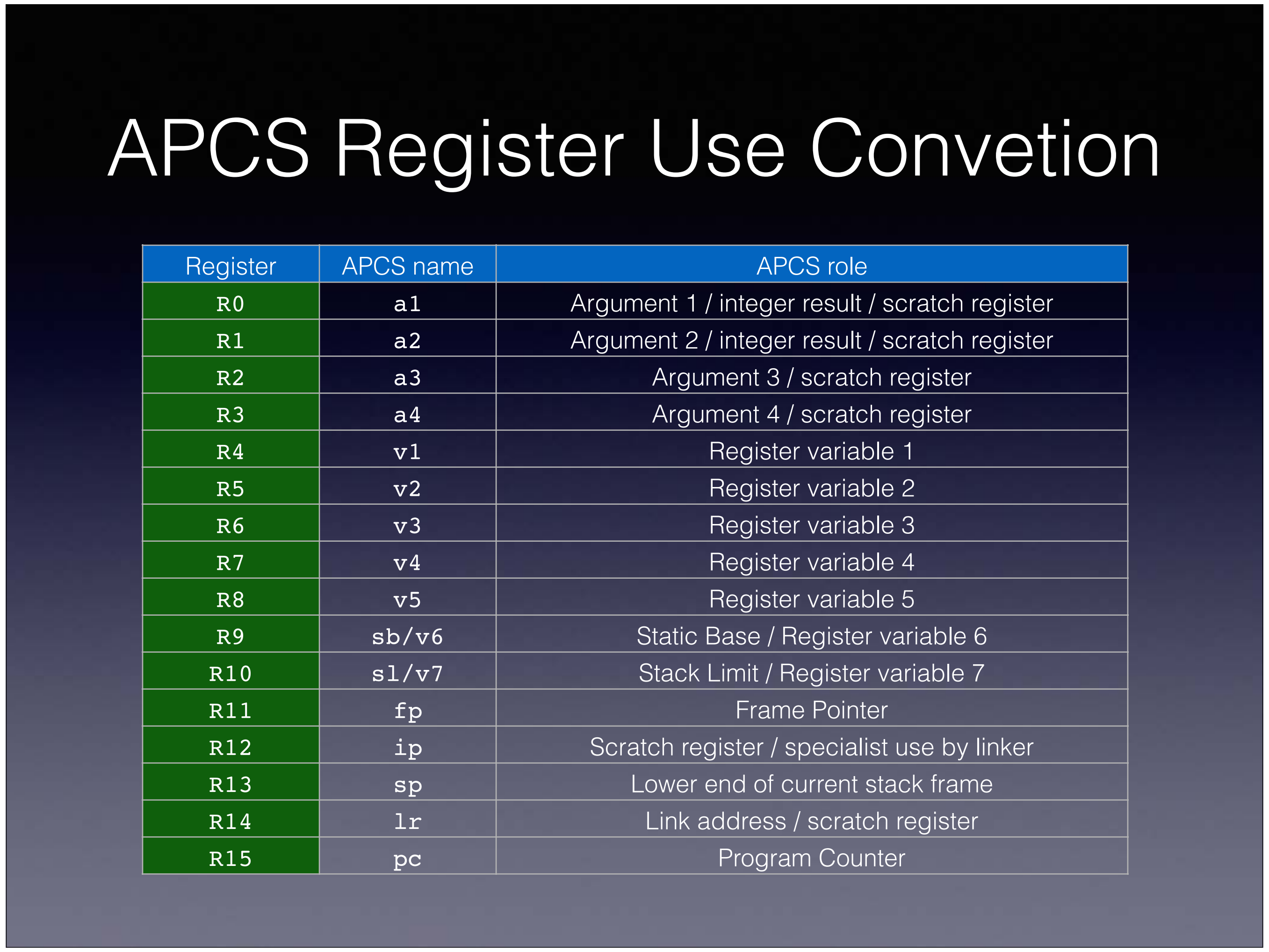
- Going to look at how we can implement a simplified grep tool
- In ARM assembler
- Will print a line if a string matches on it
- No regex at this point

Grep

- Source file will be a block of memory terminated by a null character (zero byte)
- Lines will be terminated by a newline character
- Going to need a routine to read a line
- And one to match a string in the line
    - This will be built by using a routine to compare strings

Subroutines

- Will write this as a series of sub-routines that call each other
- So we'll need to make sure we don't overwrite registers used by the calling function
- Will follow the APCS
- Will inform us as to what registers we use to store certain values



APCS Register Use Convetion

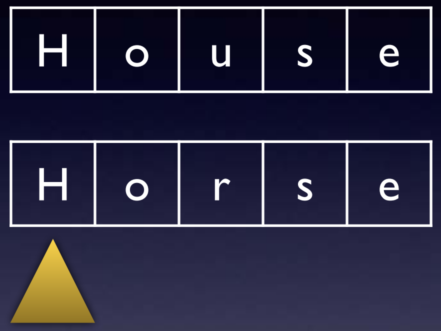
Register	APCS name	APCS role
R0	a1	Argument 1 / integer result / scratch register
R1	a2	Argument 2 / integer result / scratch register
R2	a3	Argument 3 / scratch register
R3	a4	Argument 4 / scratch register
R4	v1	Register variable 1
R5	v2	Register variable 2
R6	v3	Register variable 3
R7	v4	Register variable 4
R8	v5	Register variable 5
R9	sb/v6	Static Base / Register variable 6
R10	sl/v7	Stack Limit / Register variable 7
R11	fp	Frame Pointer
R12	ip	Scratch register / specialist use by linker
R13	sp	Lower end of current stack frame
R14	lr	Link address / scratch register
R15	pc	Program Counter

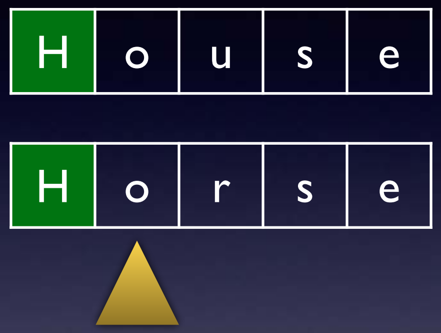
Scratch registers do not need to be preserved through a function call, but all other registers should be. As far as the caller is concerned it should be as if the function call never happened

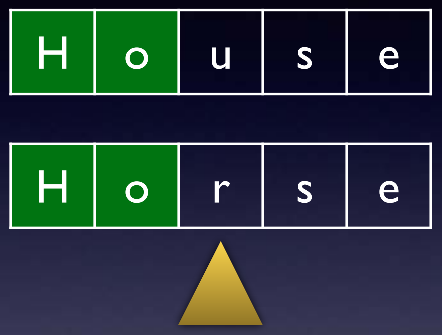
Note if more arguments are needed than registers they are placed on the stack before the procedure call. Each argument must take up a multiple of 4 bytes on the stack. For 8-byte wide values, two registers are used…

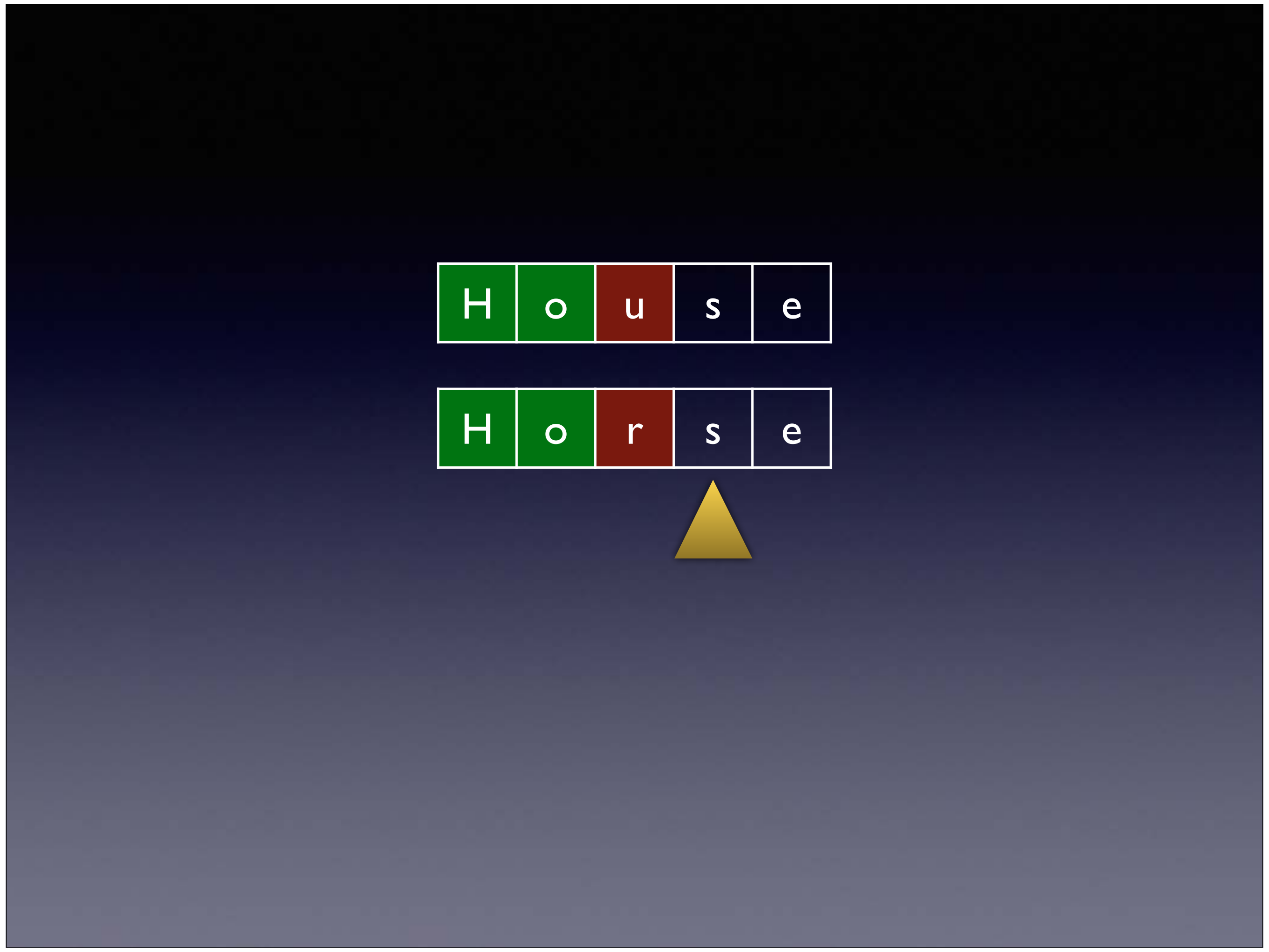
String Comparison

- Easy to compare two strings
- Just compare every character until you reach either:
  - A character that doesn’t match (therefore strings don’t match)
  - The end of the string (therefore strings match)

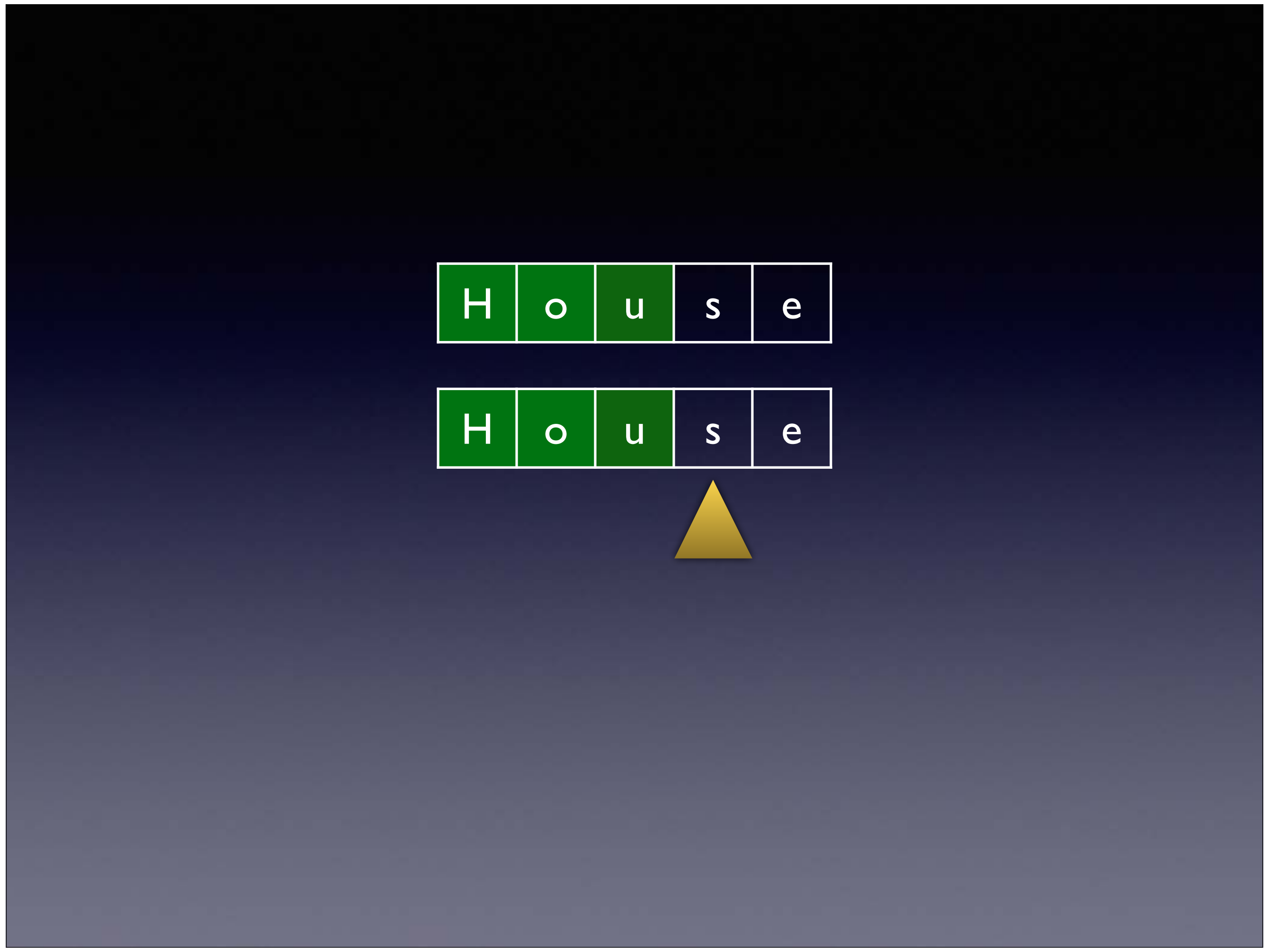

H o u s e
H o r s e


H o u s e
H o r s e

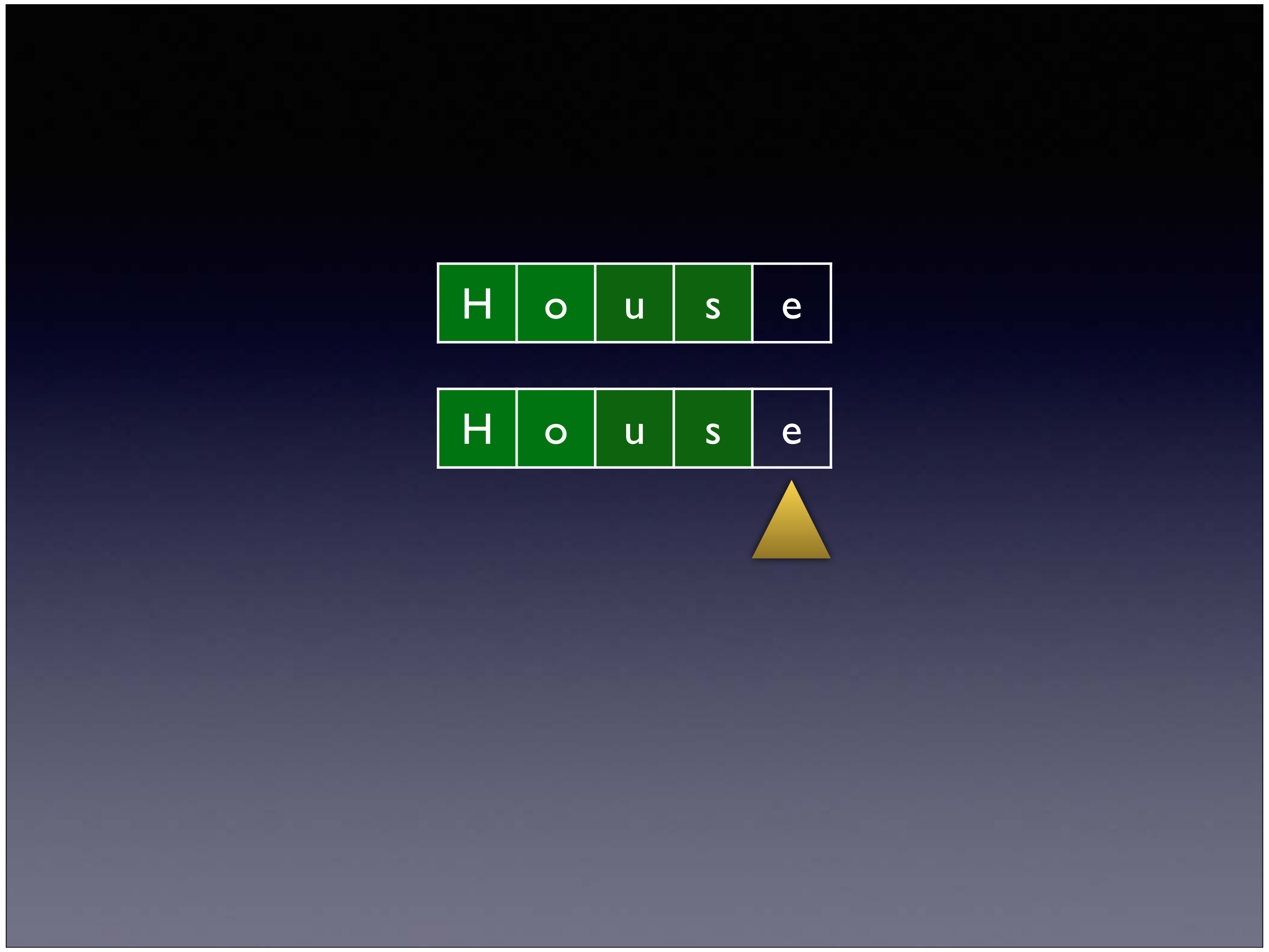

H o u s e
H o r s e



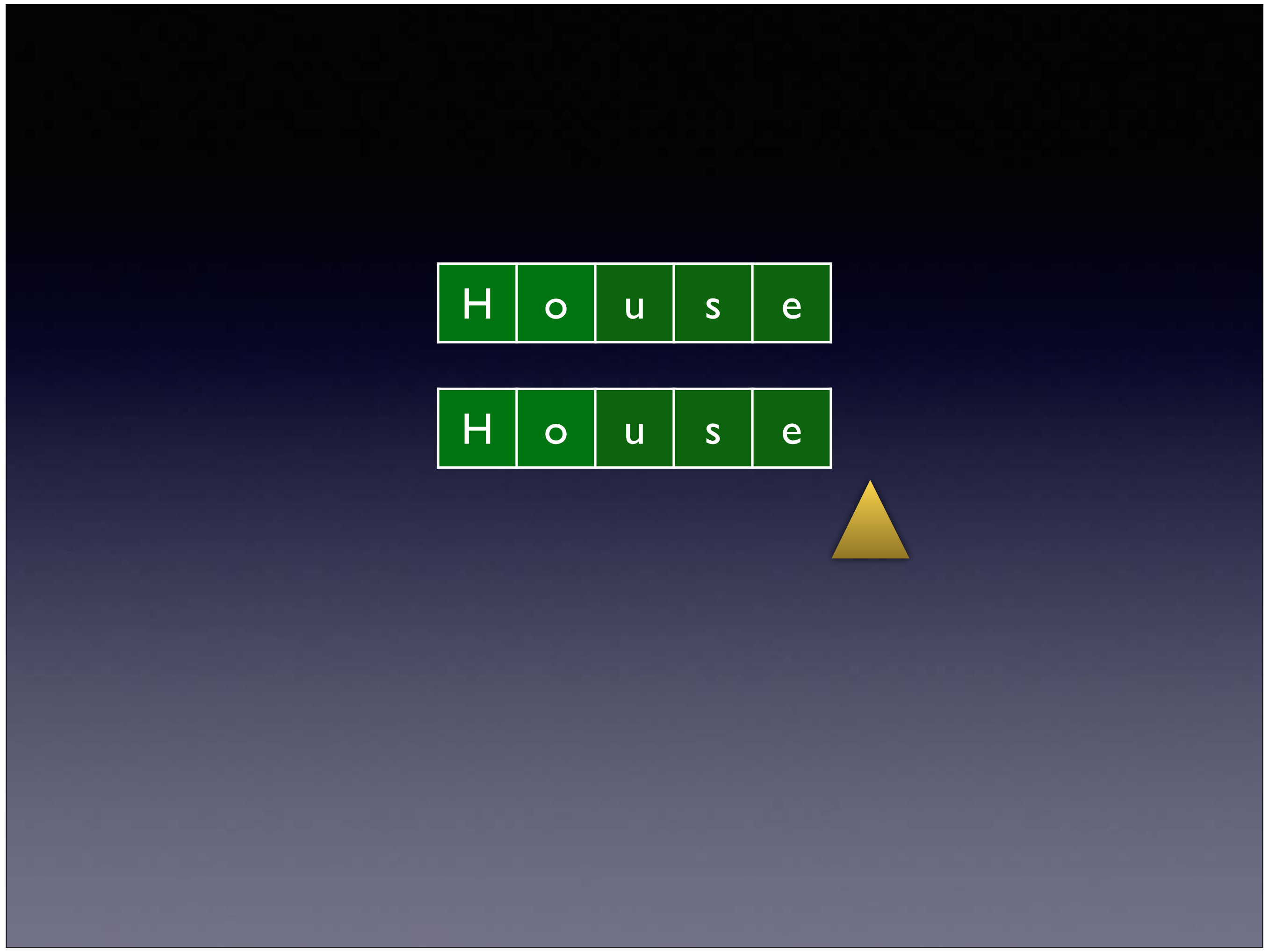
u is not the same as r so strings don’t match (obviously)



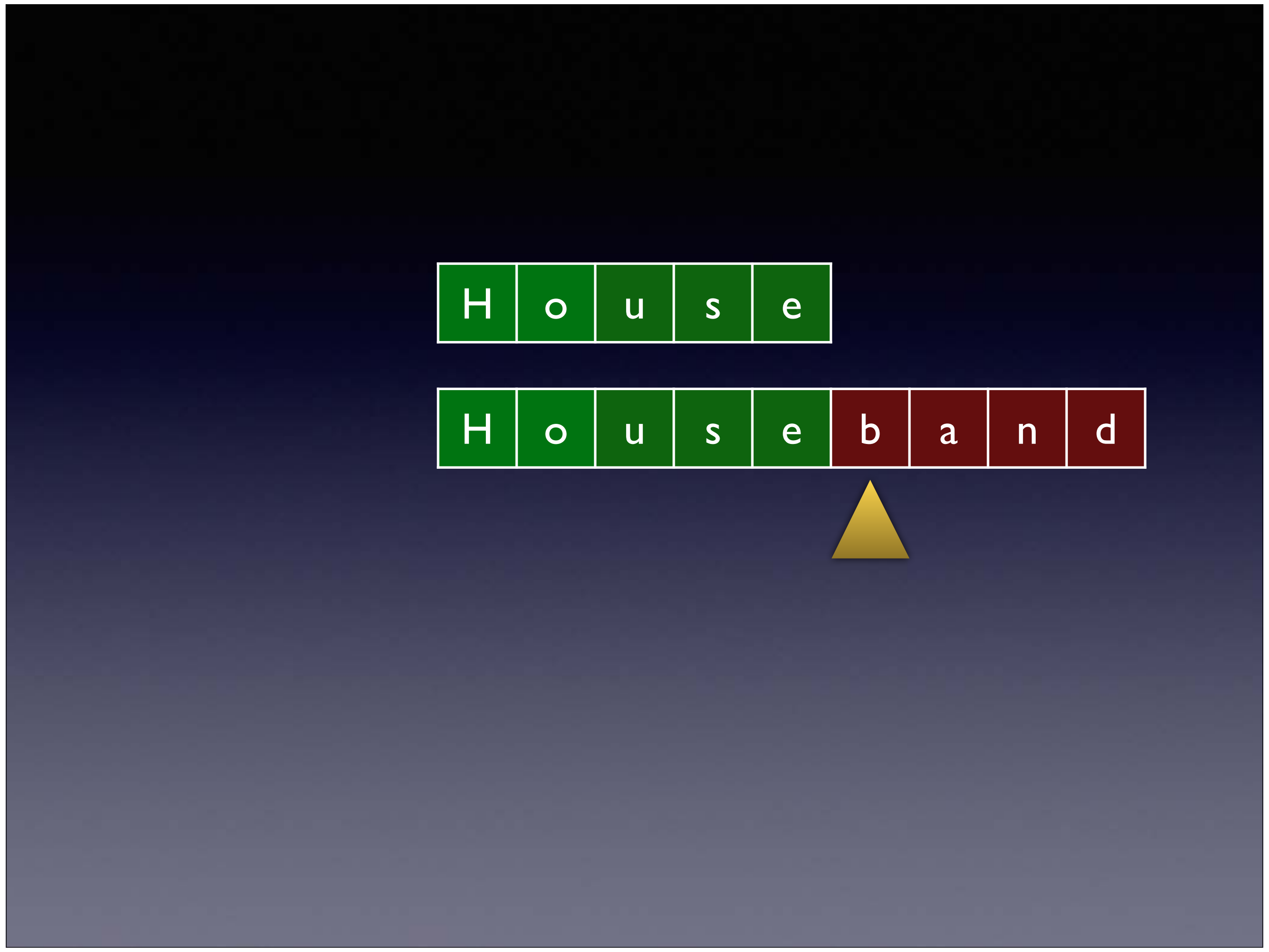
However if both strings were house they would and so we continue...



One letter left..



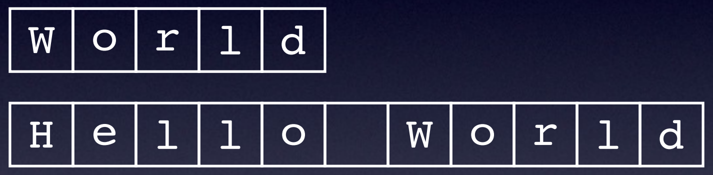
All characters match, so the strings are equal

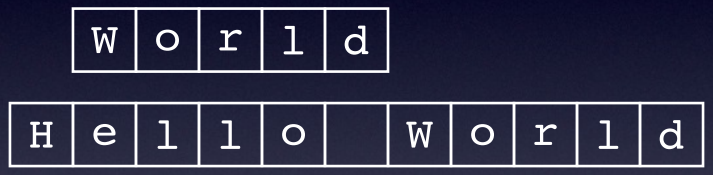


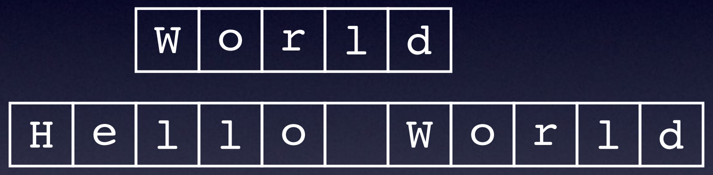
If one string is longer than the other, then obviously not equal

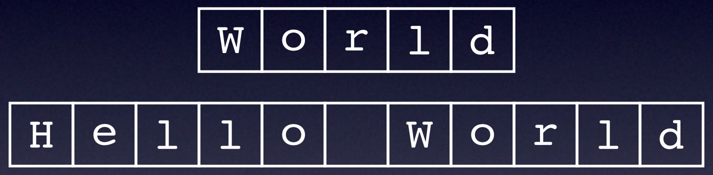
Match Within a Line

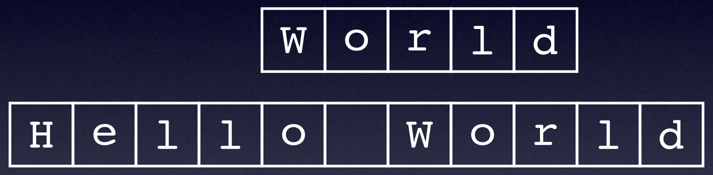
- This routine only matches a string at the beginning of the line
- We need to find it at any point in the string
- Need to make two modifications
- One string should return true if the first string is longer
- Then we need to test the string at every possible position

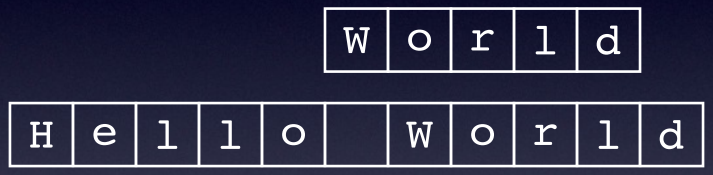

W o r l d
H e l l o   W o r l d

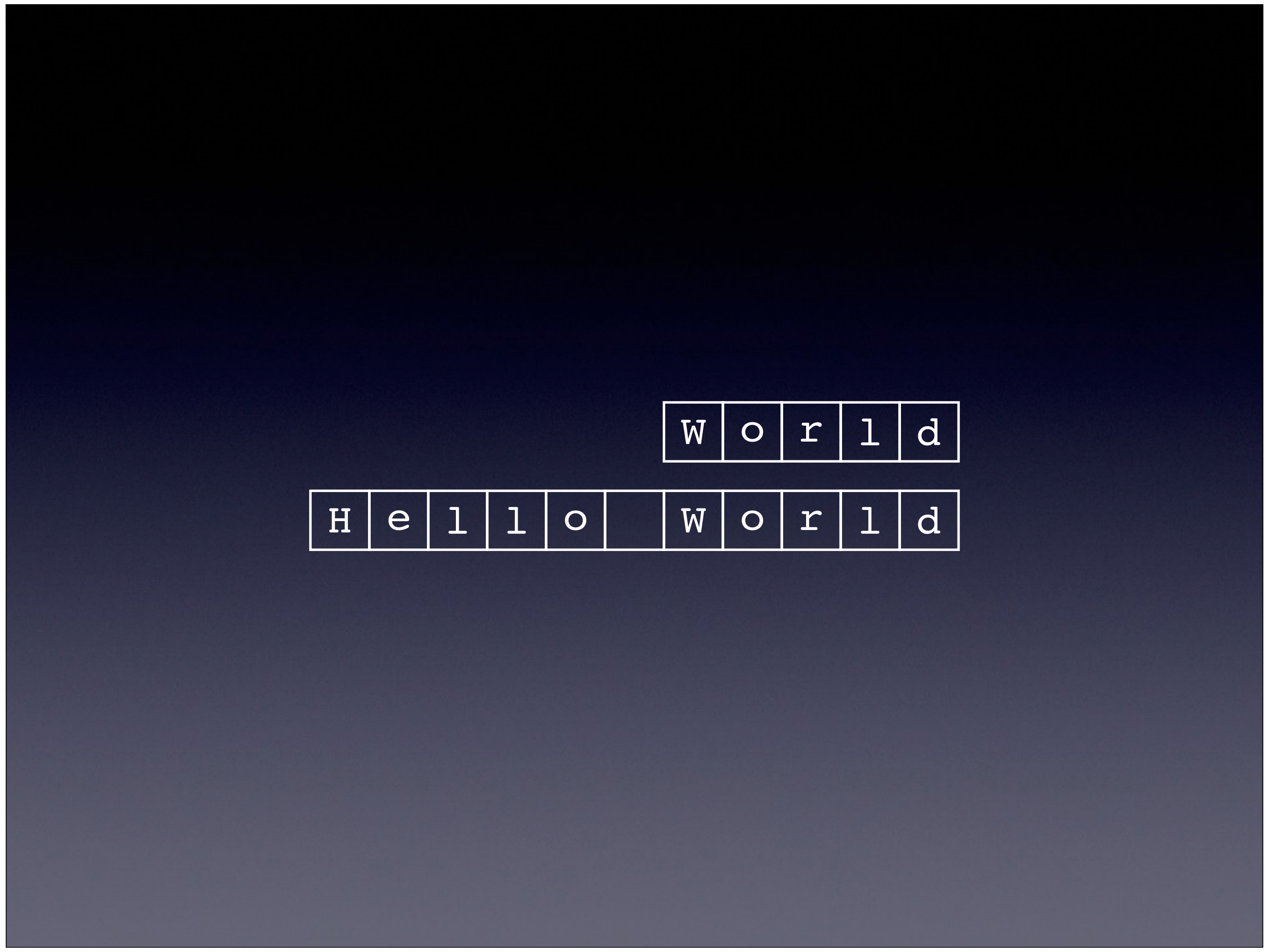

W o r l d
H e l l o   W o r l d


W o r l d
H e l l o   W o r l d


W o r l d
H e l l o   W o r l d

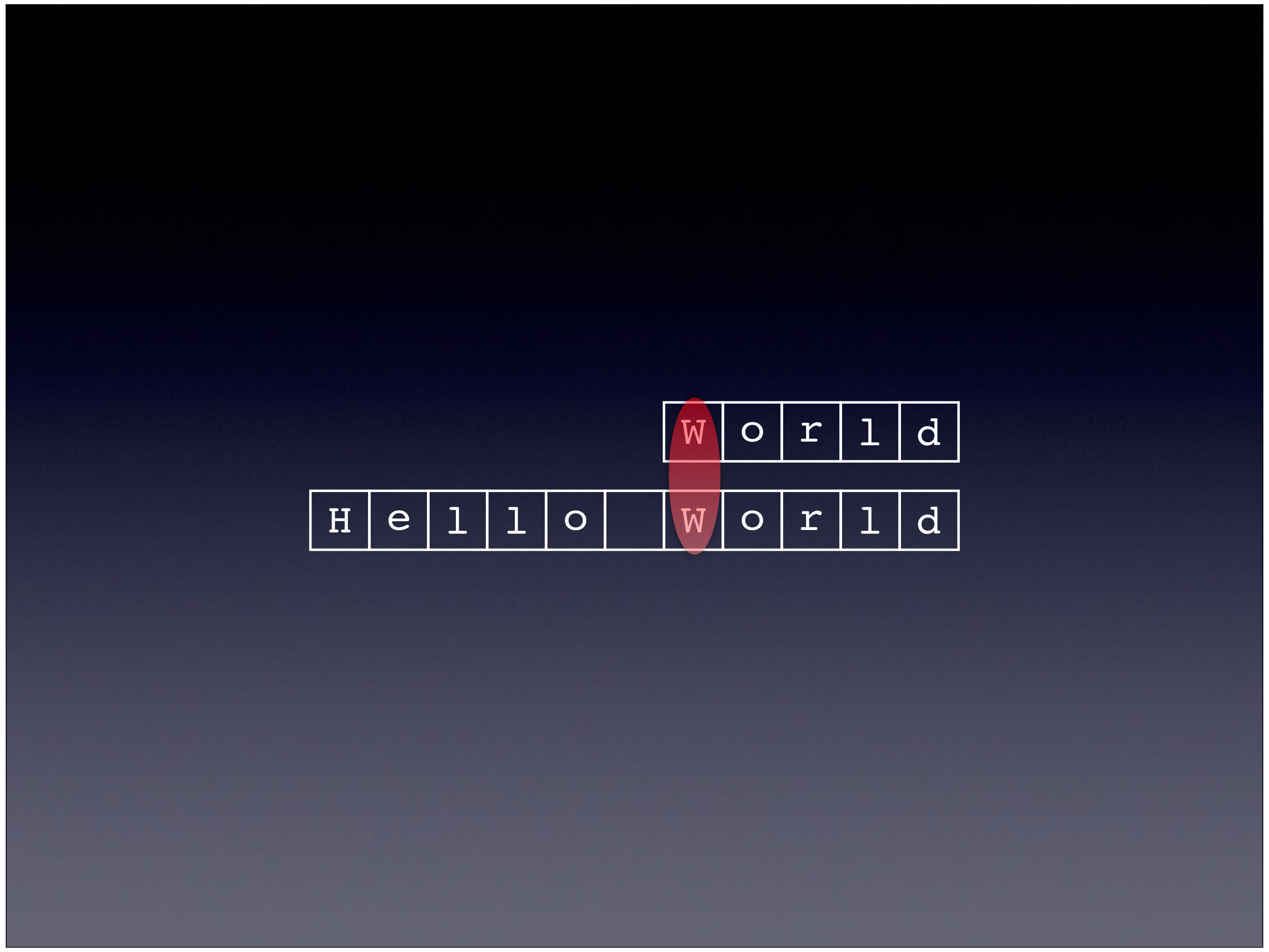

W o r l d
H e l l o   W o r l d


W o r l d
H e l l o   W o r l d



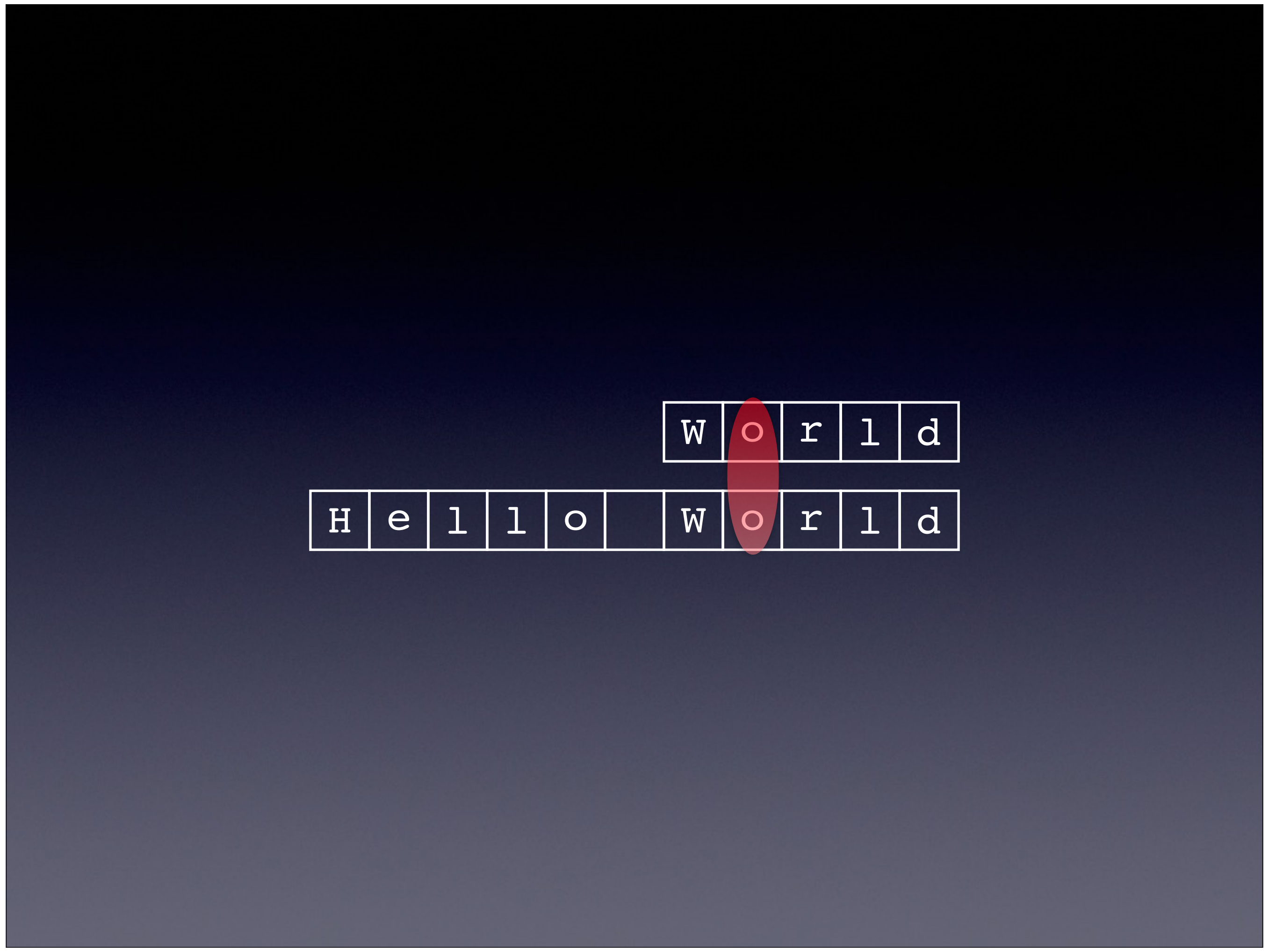
Finally we get a match

So can return true



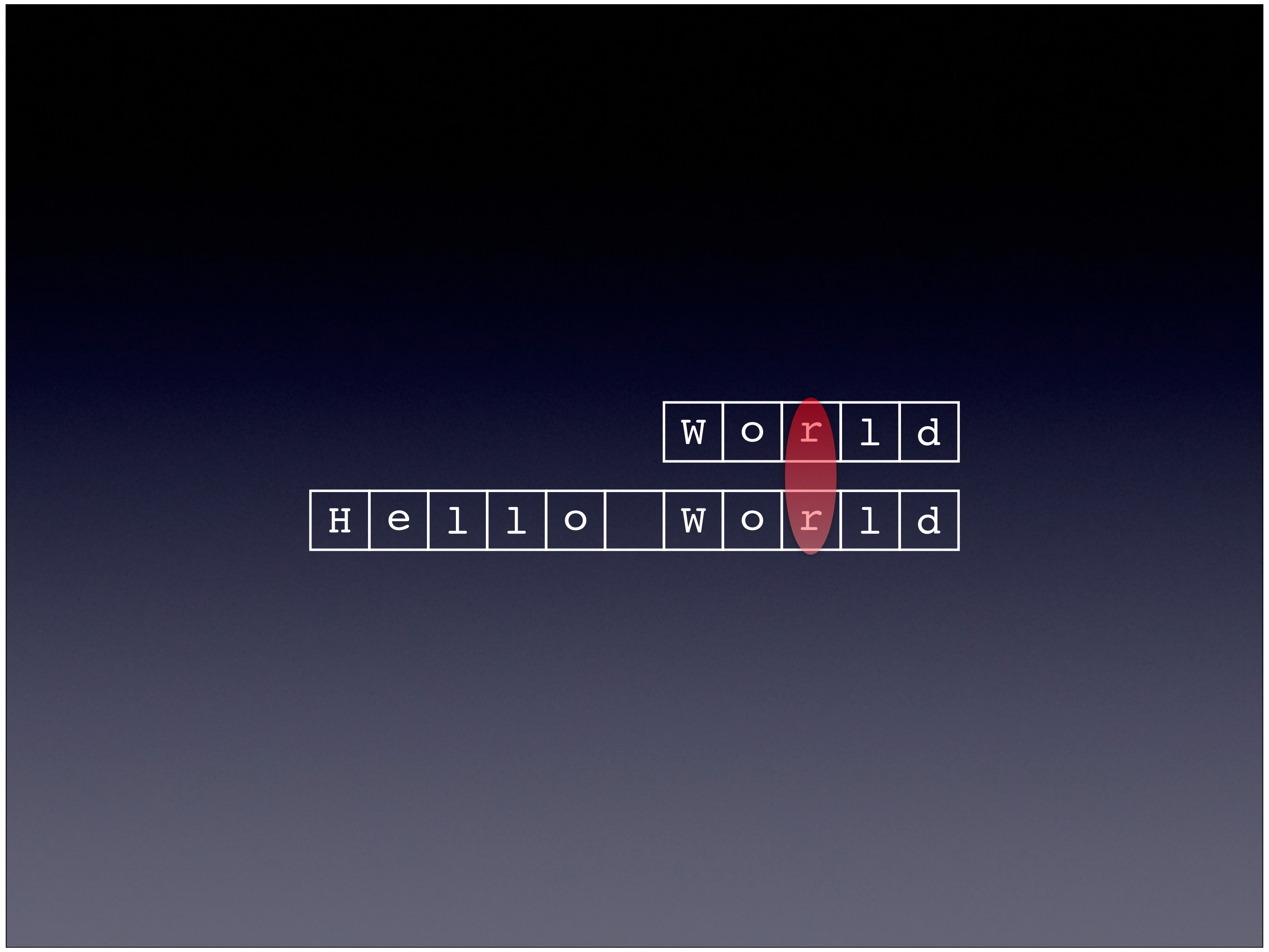
Finally we get a match

So can return true



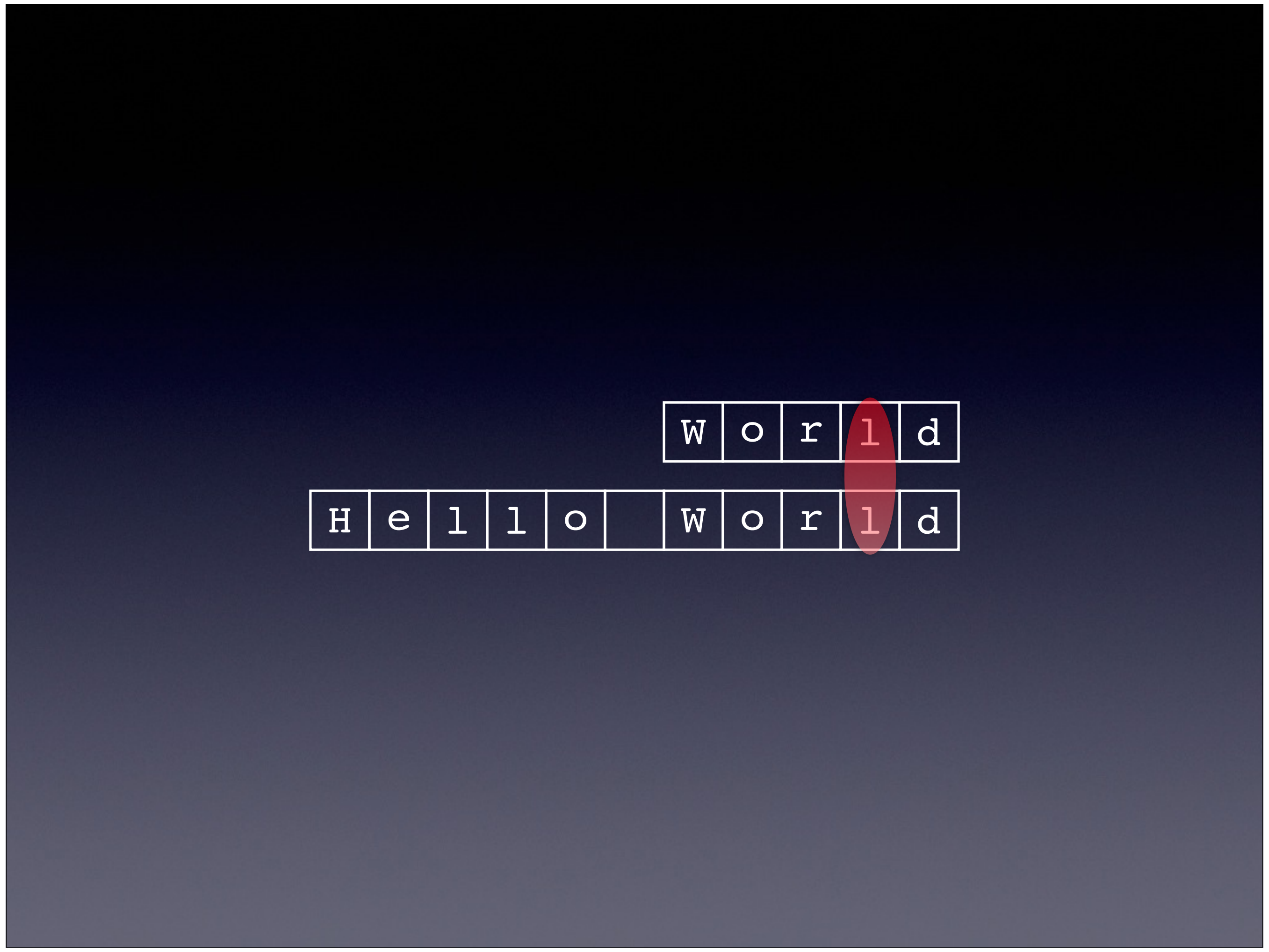
Finally we get a match

So can return true



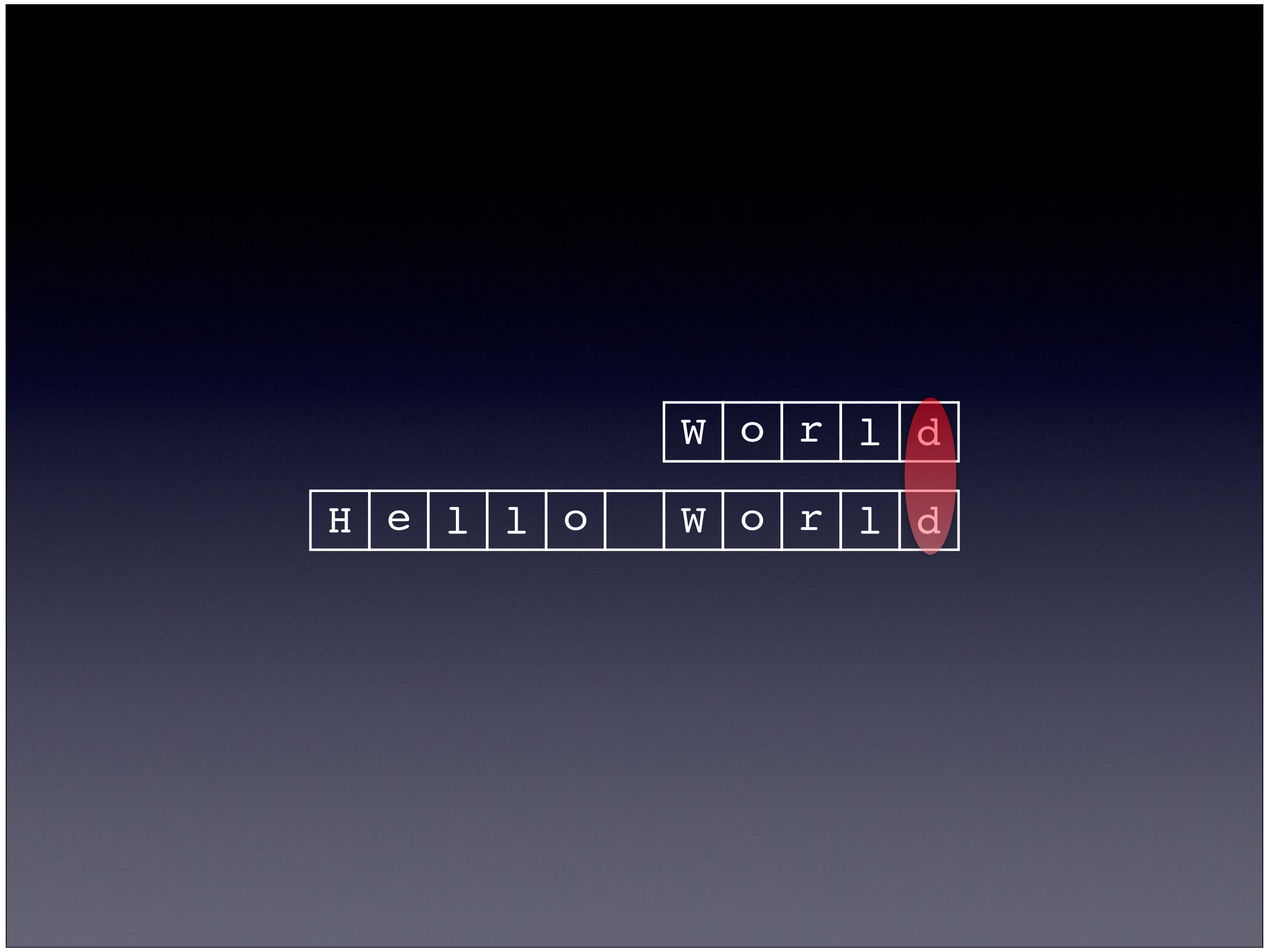
Finally we get a match

So can return true



Finally we get a match

So can return true



Finally we get a match

So can return true