# Lab Intro

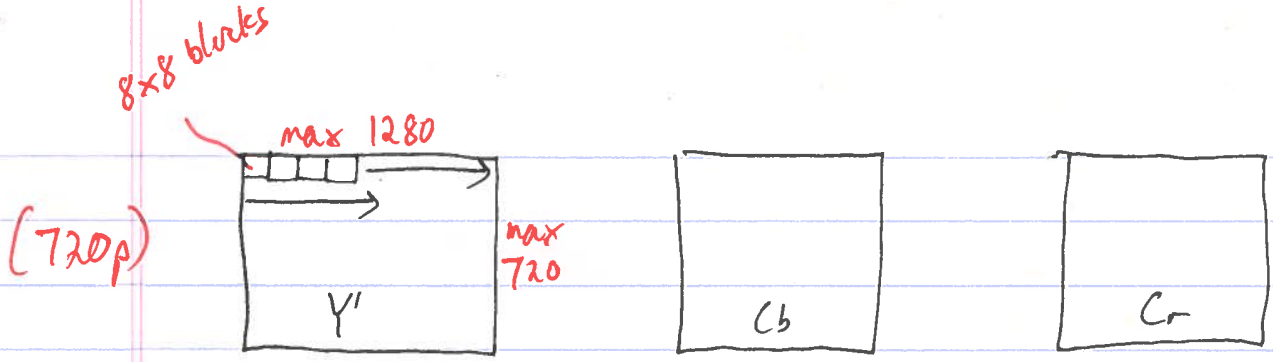
- project: video decoder on an SoC
- file format: mjpeg420
  - a sequence of frames (simplified jpeg images)

- colour spaces
  - bitmaps (.bmp files) and screen output are normally in the RGB (red green blue) space
  - a pixel can be represented using 8 bits per colour
  - RGB is hard to compress
  - our eyes are more sensitive to changes in brightness (luminance) than changes in colour (chrominance)
  - Y'CbCr space is used instead

    Y' (luminance) = 0 + 0.299R + 0.587G + 0.114B

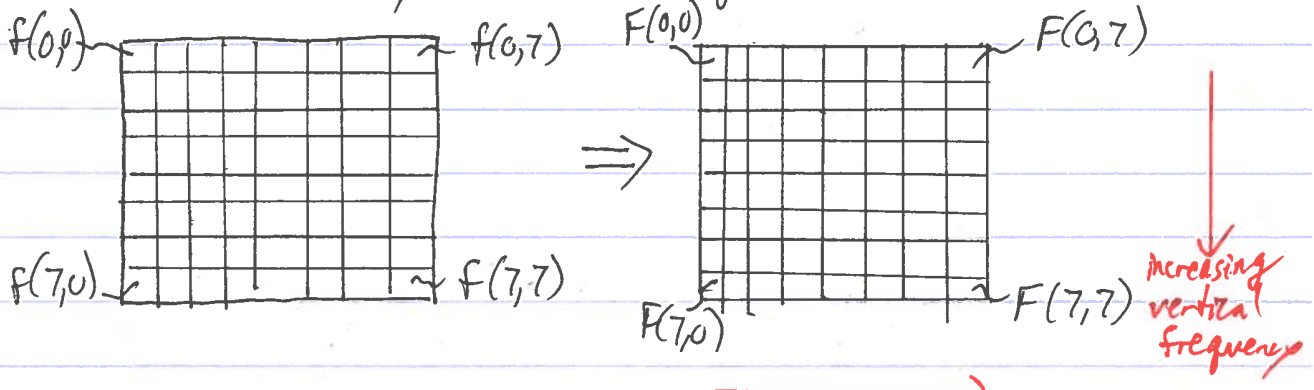
    Cb (blue chrominance) = 128 + -0.169R - 0.331G + 0.500B

    Cr (red chrominance) = 128 + 0.500R - 0.419G - 0.081R
  - we use 8 bits per channel
  - the Cb and Cr channels might be compressed more (with resulting loss of information) than the Y' channel




- encoding steps
  1) sub-divide channel into 8×8 blocks
  2) processor blocks from left to right, top to bottom
  3) apply a 2D - Discrete Cosine Transform (DCT) to each block — lossless
  4) quantize resulting 8×8 coefficients — lossy
  5) entropy encode the quantized coefficients — lossless

DCT: converts intensity data into frequency data




see computerphile Nick Pound

- first: for $x=0..7, y=0..7$
  $f(x,y) -= 128$ ($[0,255] \rightarrow [-128,127]$)

- second: for $u=0..7, v=0..7$

$$F(u,v) = \frac{1}{4} C(u) C(v) \left[ \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16} * \cos\frac{(2y+1)v\pi}{16} \right]$$

where $C(z) = \frac{1}{\sqrt{2}}$ if $z=0$
or $= 1$ if $z \neq 0$

- see Learn > Lecture > Handouts > jpeg

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 139 | 144 | 149 | 153 | 155 | 155 | 155 | 155 |
| 144 | 151 | 153 | 156 | 159 | 156 | 156 | 156 |
| 150 | 155 | 160 | 163 | 158 | 156 | 156 | 156 |
| 159 | 161 | 162 | 160 | 160 | 159 | 159 | 159 |
| 159 | 160 | 161 | 162 | 162 | 155 | 155 | 155 |
| 161 | 161 | 161 | 161 | 160 | 157 | 157 | 157 |
| 162 | 162 | 161 | 163 | 162 | 157 | 157 | 157 |
| 162 | 162 | 161 | 161 | 163 | 158 | 158 | 158 |

(a) source image samples

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 235.6 | -1.0 | -12.1 | -5.2 | 2.1 | -1.7 | -2.7 | 1.3 |
| -22.6 | -17.5 | -6.2 | -3.2 | -2.9 | -0.1 | 0.4 | -1.2 |
| -10.9 | -9.3 | -1.6 | 1.5 | 0.2 | -0.9 | -0.6 | -0.1 |
| -7.1 | -1.9 | 0.2 | 1.5 | 0.9 | -0.1 | 0.0 | 0.3 |
| -0.6 | -0.8 | 1.5 | 1.6 | -0.1 | -0.7 | 0.6 | 1.3 |
| 1.8 | -0.2 | 1.6 | -0.3 | -0.8 | 1.5 | 1.0 | -1.0 |
| -1.3 | -0.4 | -0.3 | -1.5 | -0.5 | 1.7 | 1.1 | -0.8 |
| -2.6 | 1.6 | -3.8 | -1.8 | 1.9 | 1.2 | -0.6 | -0.4 |

(b) forward DCT coefficients

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | 11 | 10 | 16 | 24 | 40 | 51 | 61 |
| 12 | 12 | 14 | 19 | 26 | 58 | 60 | 55 |
| 14 | 13 | 16 | 24 | 40 | 57 | 69 | 56 |
| 14 | 17 | 22 | 29 | 51 | 87 | 80 | 62 |
| 18 | 22 | 37 | 56 | 68 | 109 | 103 | 77 |
| 24 | 35 | 55 | 64 | 81 | 104 | 113 | 92 |
| 49 | 64 | 78 | 87 | 103 | 121 | 120 | 101 |
| 72 | 92 | 95 | 98 | 112 | 100 | 103 | 99 |

(c) quantization table

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 15 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| -2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(d) normalized quantized coefficients

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 240 | 0 | -10 | 0 | 0 | 0 | 0 | 0 |
| -24 | -12 | 0 | 0 | 0 | 0 | 0 | 0 |
| -14 | -13 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(e) denormalized quantized coefficients

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 144 | 146 | 149 | 152 | 154 | 156 | 156 | 156 |
| 148 | 150 | 152 | 154 | 156 | 156 | 156 | 156 |
| 155 | 156 | 157 | 158 | 158 | 157 | 156 | 155 |
| 160 | 161 | 161 | 162 | 161 | 159 | 157 | 155 |
| 163 | 163 | 164 | 163 | 162 | 160 | 158 | 156 |
| 163 | 164 | 164 | 164 | 162 | 160 | 158 | 157 |
| 160 | 161 | 162 | 162 | 162 | 161 | 159 | 158 |
| 158 | 159 | 161 | 161 | 162 | 161 | 159 | 158 |

(f) reconstructed image samples

Figure 10. DCT and Quantization Examples

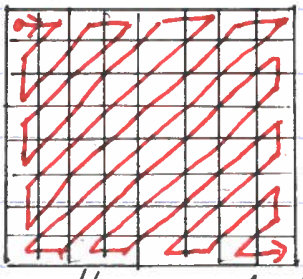
Gregory Wallace. The JPEG Still Picture Compression Standard in IEEE Transactions on Consumer Electronics, February, 1992.

- F(0,0) is the DC coefficient (zero frequency) and the rest of F(u,v) values are the AC coefficients
- F(0,0) represents average f(x,y) scaled by 8 ([-1024, +1023]) 11 bits
- the more significant values tend to the upper left (lower frequencies) because images tend to vary slow across an 8x8 block

Quantization (compression)
- each coefficient F(u,v) is divided by the corresponding value from an 8x8 quantization table and rounded to the nearest integer
- our eyes are less perceptive of high frequency changes so the lower right is quantized more aggressively which tends to produce lots of zeroes
- quantization is lossy
- image quality and compression can be varied by using different tables
- one table is used for the Y' channel and another for the Cb and Cr channels

# Entropy Encoding

- compresses each block losslessly
- coefficients are scanned in zig-zag order



- this order typically yields long strings of zeros
- DC coefficients tend to change slowly between blocks so are encoded as the difference with the DC coeff. of the previous block

- coefficients are represented as Variable Length Integers (VLI)

| amplitude | size (in bits) |
|---|---|
| -1, 1 | 1 |
| -3, -2, 2, 3 | 2 |
| -7.. -4, 4.. 7 | 3 |
| -15.. -8, 8.. 15 | 4 |
| ... | |
| -1023.. -512, 512..1023 | 10 |

(Wallace '92 Table 3)

(DC coeff has twice range of AC coeff) ⇒ 11 bits

- DC coeff. representation: (SIZE) (AMPLITUDE)
  4 bits
- AC coeff. representation: (RUNLENGTH, SIZE)(AMPLITUDE)
  # of preceding zeros — 4 bits — 4 bits — # of non-zero AC coeff.

- example (from handout image d)

DC coeff.

$$\begin{bmatrix} 15 & 0 & -1 & 0 & \cdots \\ -2 & -1 & 0 & 0 & \cdots \\ -1 & -1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

- suppose $DC_{i-1} = 12$, then $\Delta DC_i = +3$.
(2)(3), (1,2)(-2), (0,1)(-1), (0,1)(-1), (0,1)(-1), (2,1)(-1), (0,0)

↑ EOB (end-of-block escape)

- (15,0) ← ZRL (represents 16 zeros)

e.g. $\underbrace{0\,0\,..\,0}_{20}\ 5 \Rightarrow (15,0)(4,3)(5)$

Frame Types

- mjpeg423 takes advantage of the similarity between successive frames
- two frame types: Index (I-), Progressive (P-)
- I-frame: stored as a normal jpeg image
  - DC coefficient is differential between blocks

$$\Delta DC_i = DC_i - DC_{i-1}$$ (block #)

- P-frame: differential encoding of DC and AC coefficients between frames

(frame #)

$$\Delta DC_i^j = DC_i^j - DC_i^{j-1} \qquad \Delta AC_i^j(x,y) = AC_i^j(x,y) - AC_i^{j-1}(x,y)$$

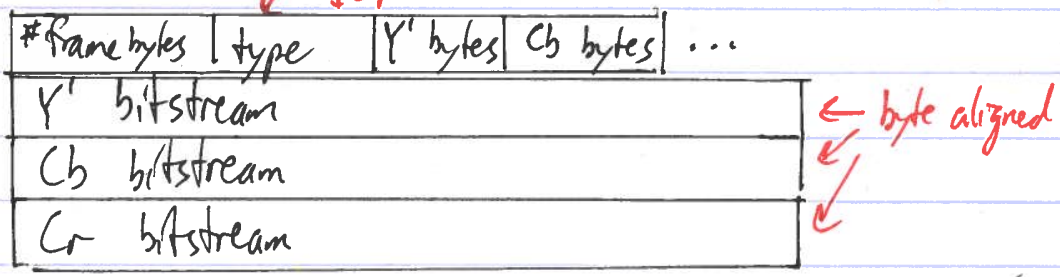
# File Format


- Header

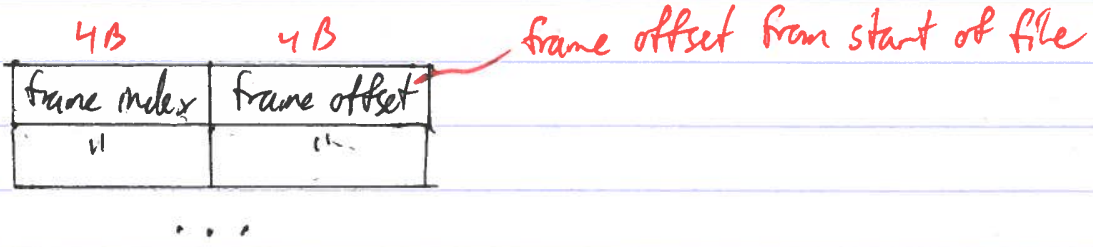

total frames (I- and P-)

each 4B

- Payload - sequence of frames (4B aligned)




- Trailer: size = 8 * # i-frames



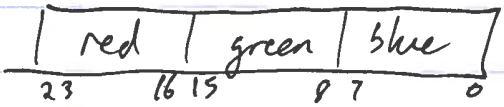

# Decode Steps

1) lossless entropy decode
2) dequantize block (multiply by table values)
3) apply inverse DCT (IDCT)

for $x = 0..7, y = 0..7$

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v) * \cos\frac{(2x+1)u\pi}{16} * \cos\frac{(2y+1)v\pi}{16}\right]$$

Given a reference implementation
- does encode and decode
- decoder outputs a sequence (one per frame) of bmp files using 32 bits/pixel
- VDMA transfers 24 bits/pixel




- overall goal: achieve 24 fps playback