representative of
student view of class:
(from midterm evaluation)

it's supposed
to be

3710: fast, confusing, profane, different languages,
and yet somehow satisfying
(also, odd guy at the front [stylishly dressed])

my response to the midterm evaluations:

midterm evaluation: likes

- even after this class, you still want to be engineers
- most of you find the class difficult/challenging, in a good way (masochists, I guess)
- real-world/hands-on experience appreciated

midterm evaluation: dislikes/areas for improvement

- time (number one complaint)
  - too many topics/moves too fast (want two semesters)
    - cannot exceed state-mandated #credits; essentially the same number of topics, lab wise, as previous semesters (but cover more material in lecture)
- hw (5--7 hrs); lab (8--15)
  - 3+ hours outside of lecture/lab expected per credit hour in engineering (you != business majors); intensive course
- too early

midterm evaluation: dislikes/areas for improvement

- lab
  - TAs not helpful
    - Bring specific cases to me
    - instructed not to give out answers; they are not instructors but facilitators (they want to help you find answers on your own); share their experience; help to avoid common problems (frying ports, etc)
  - lab instructions lack clarity
    - agreed (we can do better and will); these are revised labs (three boards in three semesters): need your help to point things out
    - suggest useful sections of datasheet on wiki for each lab (will be incorporated)

midterm evaluation: dislikes/areas for improvement

- lab
  - board isn't the same in lecture as lab
    - this is intentional: need the experience of going through datasheet (harsh learning curve: try to configure lecture board w/o lecture notes for practice)
    - if the same: I do all of the configuration for you (learning is not painless); will post pointers for things like pin muxing, though
  - datasheet woes
    - Necessary skill (according to industry partners)
    - Always check init. and config. sections; overview; LCD: posted working drivers
    - Labs try to ease you into them; open to suggestions

midterm evaluation: dislikes/areas for improvement

- more examples
  - always; different from those in book; challenge examples
  - write them in class: willing to, if we have two hours per lecture
- homework clarity
  - it's clear to me (grumble about ambiguous customer specifications here); please let me know and will disseminate clarifications
  - late clarifications: if I only learn it's not clear the day before it's due, can only clarify the day before it's due
  - more office hours: usually alone on Tuesdays and open door policy (usually answer even when door is closed)

midterm evaluation: dislikes/areas for improvement

- tutorials/external resources
  - Per syllabus, free tutoring service (if not helpful, let me know)
  - can put resources on wiki (if you find something useful, add it---that's what I do)
- hw/exam/lab difficulty w/time constraints
  - only way I know to prepare you for final project (which I want you to use to demonstrate your skills to employers)
  - like what you will face in your job (won't just be asked to go to look something up)

midterm eval: final thoughts

you should never
feel like:

midterm eval: final thoughts

know that understanding takes time:

structs

```
struct SP
{
    char a;
    int b;
};
```

```
typedef struct
{
    char a;
    int b;
}SP;
```

use:

```
struct SP z;
struct SP *zp;
z.a = 0xFF;
zp = &z;
zp->b = 0xABCD
```

use:

```
SP z;
SP *zp;
z.a = 0xFF;
zp = &z;
zp->b = 0xABCD
```

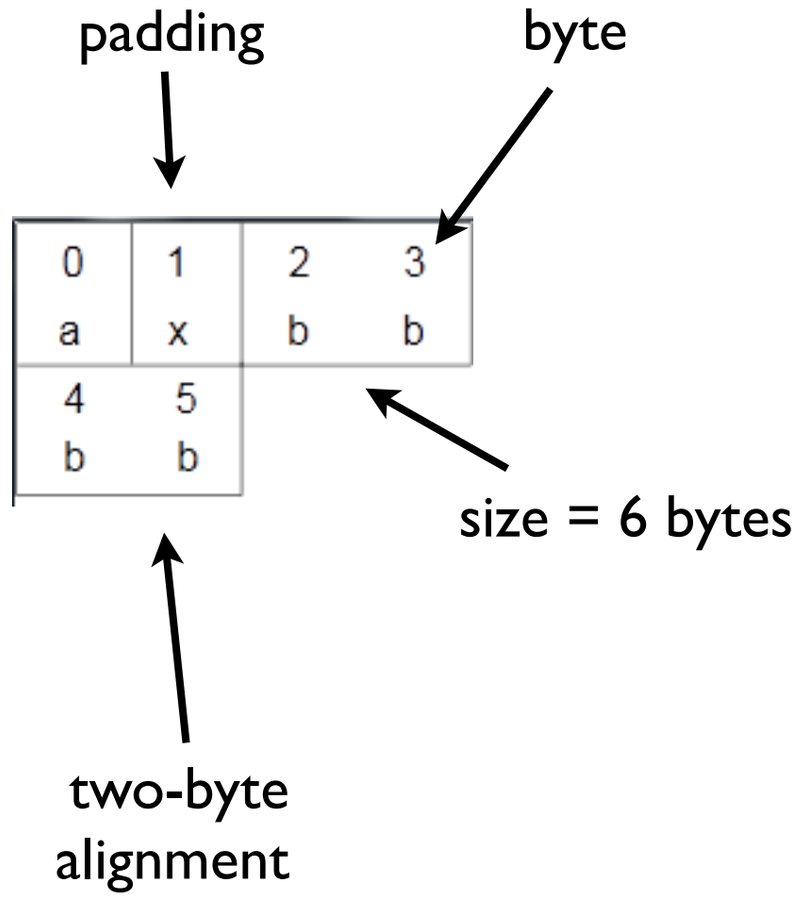
structs: memory allotment

```
struct SP
{
    char a;
    int b;
};
```

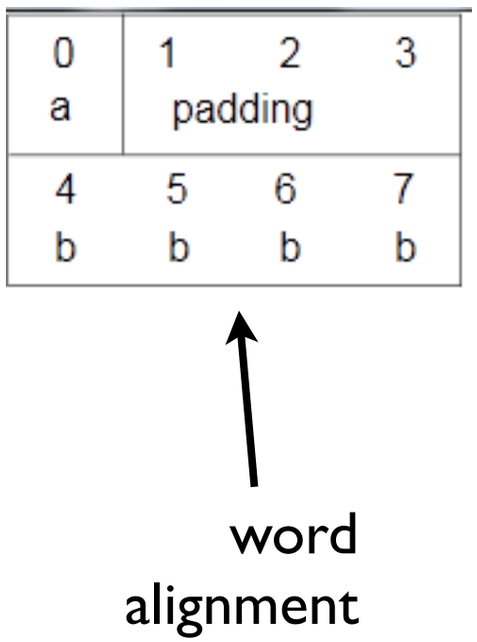
nominal bytes = 5

packed

#pragma pack(2)



non-packed



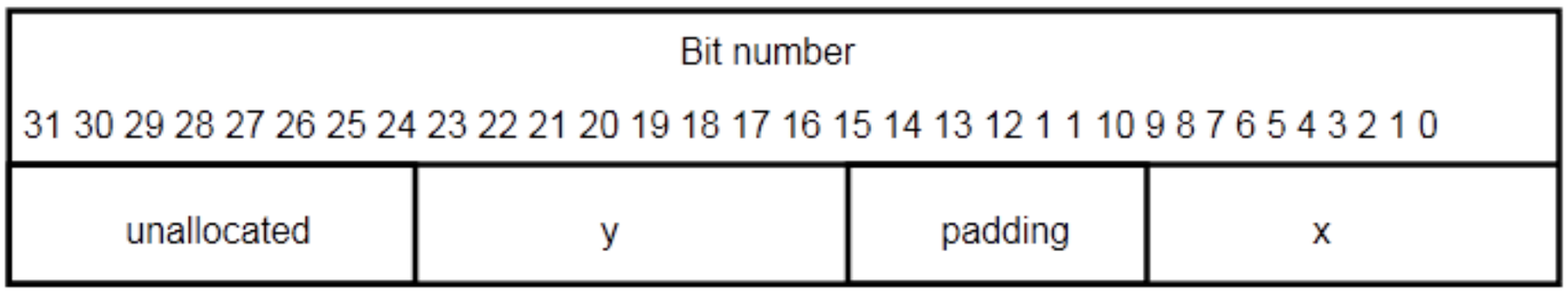
bitfields

```
struct X
{
    int x:10;
    char y:8;
};
```

allocate ten bits for int

allocate eight bits for char

results in



data must be byte-aligned

ways to access a register var

access each bit:

```
typedef struct{
   volatile unsigned int _0:1,_1:1,_2:1,_3:1,_4:1,_5:1,_6:1,_7:1,
   _8:1,_9:1,_10:1,_11:1,_12:1,_13:1,_14:1,_15:1,
   _16:1,_17:1,_18:1,_19:1,_20:1,_21:1,_22:1,_23:1,
   _24:1,_25:1,_26:1,_27:1,_28:1,_29:1,_30:1,_31:1;
}bitReg;
```

access each half-word:

```
typedef struct{
   volatile unsigned short _0,_1;
}halfwordReg;
```

access each byte:

```
typedef struct{
   volatile unsigned char _0,_1,_2,_3;
}byteReg;
```

ways to access a register var

access each bit:

```
bitReg bREG;
bREG._17 = 0x1;
```
← bit 17 = 1

access each half-word:

```
halfwordReg HREG;
HREG._1 = 0xAABB;
```
↖ upper 16 bits = 0xAABB

access each byte:

```
byteReg BREG;
BREG._2 = 0xCD;
```
← bits [23:16]= 0xCD

combing ways to access a register var

```
typedef union{
  bitReg b;
  byteReg B;
  halfwordReg H;
  volatile unsigned int W;
}REG;
```

all these sit at the same memory location

e.g. set bit one to 1:

```
REG R;
R.b._1 = 1;
R.B._0 = 0x2;
R.H._0 = 0x2;
R.W = 0x2;
```

ex: enable clock for PD

ptr of type REG pointing to

```
REG *SYSCTL = (REG *) 0x400FE108;
SYSCTL->b._3 = 0x1; //PD enable @ bit 3
```

b/c ptr

struct for GPIO peripheral

```
typedef struct{
   volatile unsigned char DATA_[1019]; //TI bit addresses
   REG DATA; //0x3FC
   REG DIR; //0x400
   REG IS; //0x404
   REG IBE; //0x408
   REG IEV; //0x40C
   REG IM; //0x410
   REG RIS; //0x414
   REG MIS; //0x418
   REG ICR; //0x41C
   REG AFSEL; //0x420
    ...
```

put reg at
correct offset

ex: GPIO on PD

(PD.0 input, PD[7:1] output)

structs defined here

point to PD base

```
#include "GPIO.h"
GPIO *PD = (GPIO *) 0x40007000;
REG *SYSCTL = (REG *) 0x400FE108;

int main(void)
{
   volatile unsigned int PD0;

   SYSCTL->b._3 = 0x1; //enable peripheral clock

   PD->DIR.B._0 = 0xFF; //all pins as output
   PD->DIR.b._0 = 0x0;  //pin zero as input
   PD->DEN.B._0 = 0xFF; //enable all pins (PD[7:1] as output)

   //output 0xAA
   PD->DATA.B._0 = 0xAA;

   //get input
   while(1)
      PD0 = PD->DATA.b._0;
}
```