Meet & Greet!

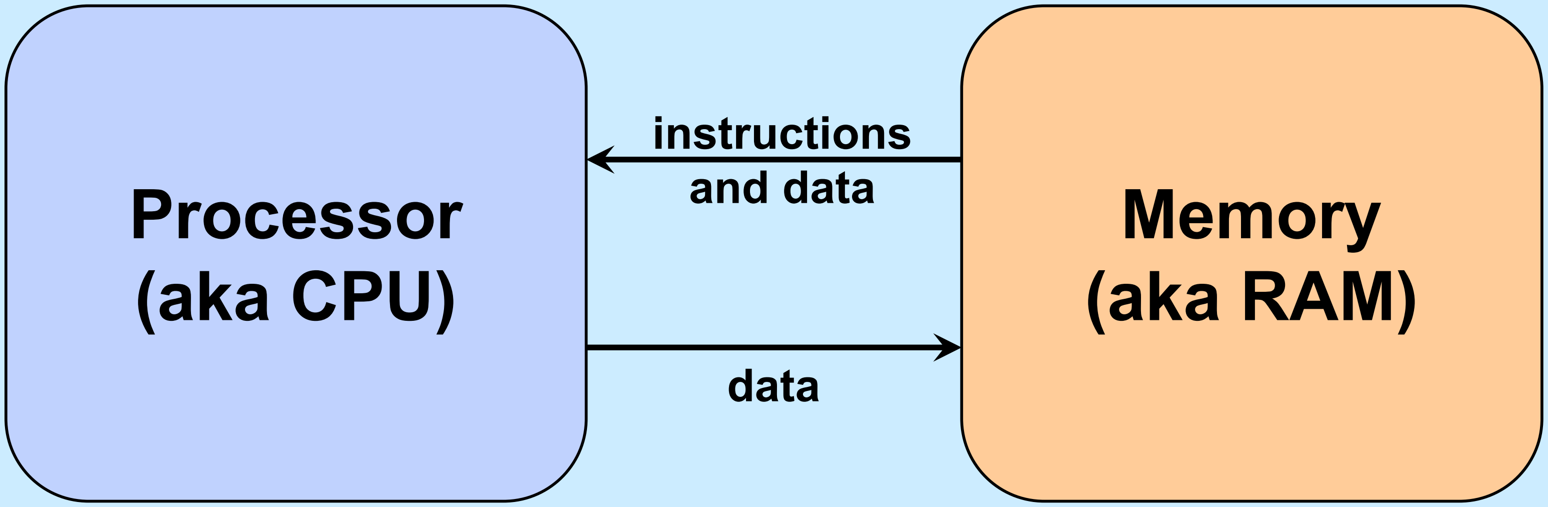
Come hang out with your TAs and
Fellow Students
(& eat free insomnia cookies)

When : Friday, Sept. 29th. 5-6 pm
Where : 3rd Floor Atrium, CIT

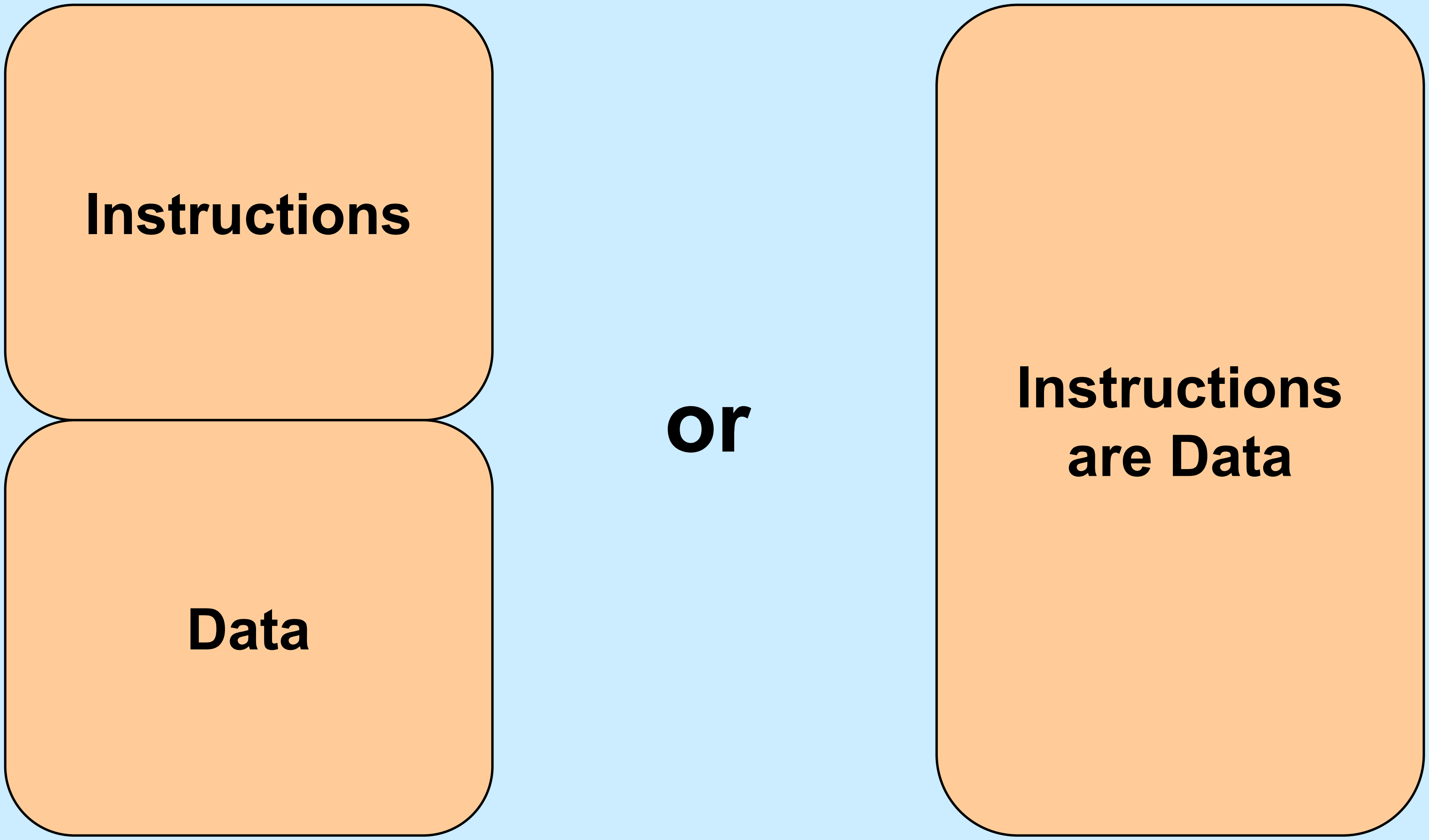
CS 33

Intro to Machine Programming

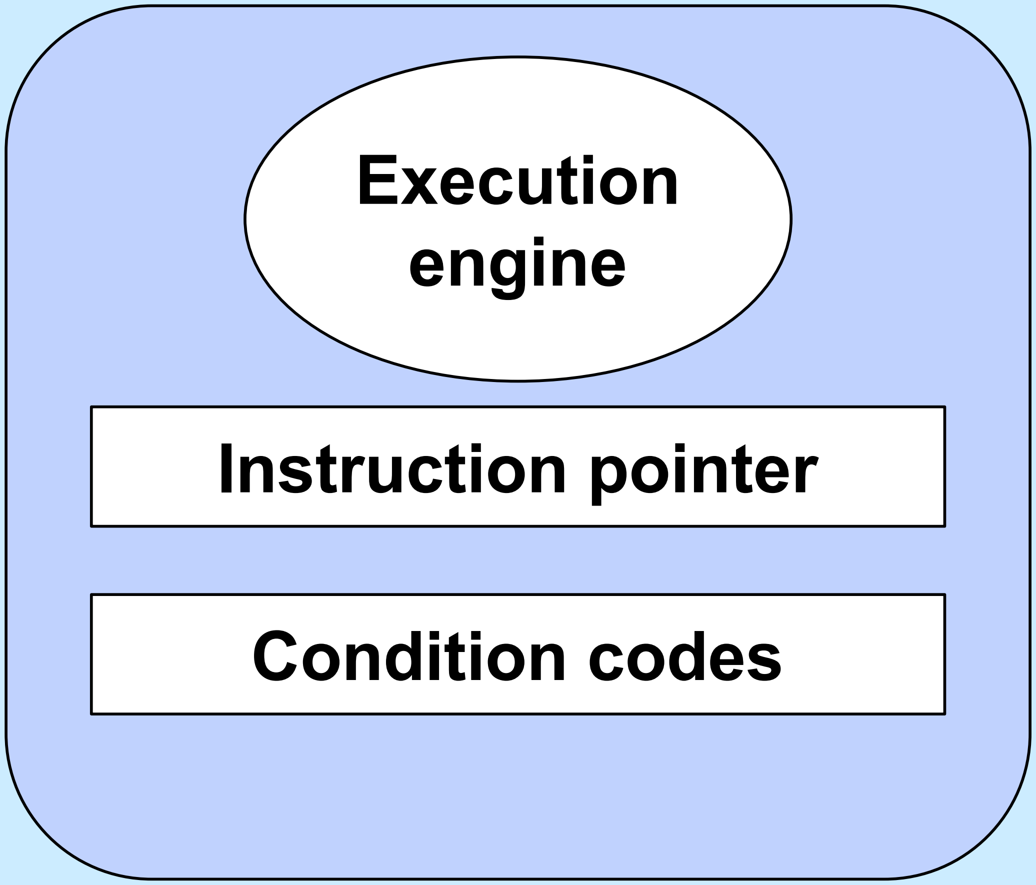
Machine Model



Memory



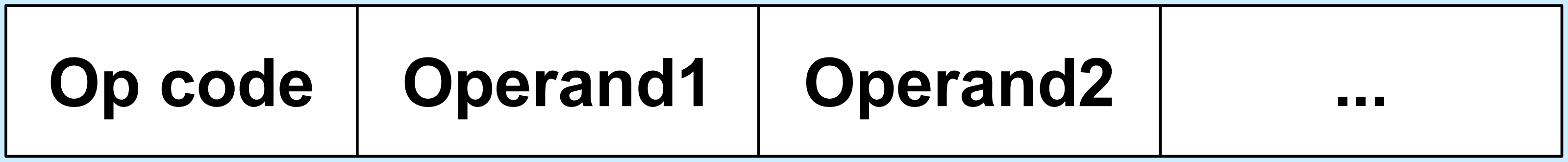
Processor: Some Details



Processor: Basic Operation

```
while (forever) {
  fetch instruction IP points at
  decode instruction
  fetch operands
  execute
  store results
  update IP and condition code
}
```

Instructions ...



Operands

- Form
  - immediate vs. reference
    - » value vs. address
- How many?
  - 3
    - » add a,b,c
      - c = a + b
  - 2
    - » add a,b
      - b += a

Operands (continued)

- Accumulator
  - special memory in the processor
    - » known as a *register*
    - » fast access
  - allows single-operand instructions
    - » add a
      - acc += a
    - » add b
      - acc += b

From C to Assembler ...

```
a = (b + c) * d;

    mov    b,%acc
    add    c,%acc
    mul    d,%acc
    mov    %acc,a
```

```
if (a<b)
   c = 1;
else
   d = 1;

   cmp    a,b
   jge    .L1
   mov    $1,c
   jmp    .L2
.L1
   mov    $1,d
.L2
```

immediate operand

immediate operand

Condition Codes

- Set of flags giving status of most recent operation:
  - zero flag
    - » result was or was not zero
  - sign flag
    - » for signed arithmetic interpretation: sign bit is or is not set
  - overflow flag
    - » for signed arithmetic interpretation
  - carry flag (generated by carry or borrow out of most-significant bit)
    - » for unsigned arithmetic interpretation
- Set implicitly by arithmetic instructions
- Set explicitly by compare instruction
  - cmp a,b
    - » sets flags based on result of b-a

Quiz 1

- Set of flags giving status of most recent operation:
  - zero flag
    - » result was or was not zero
  - sign flag
    - » for signed arithmetic interpretation: sign bit is or is not set
  - overflow flag
    - » for signed arithmetic interpretation
  - carry flag (generated by carry or borrow out of most-significant bit)
    - » for unsigned arithmetic interpretation
- Set explicitly by compare instruction
  - cmp a,b
    - » sets flags based on result of b-a

Which flags are set to one by "cmp 2,1"?

a) overflow flag only
b) carry flag only
c) sign and carry flags only
d) sign and overflow flags only
e) sign, overflow, and carry flags

Jump Instructions

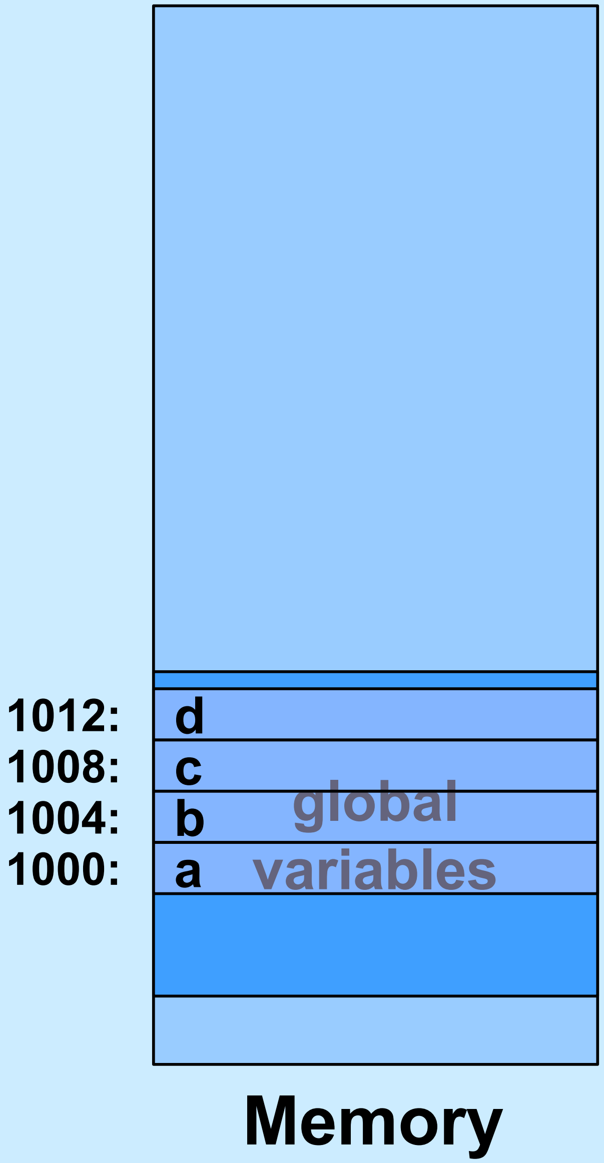
- Unconditional jump
  - – just do it
- Conditional jump
  - – to jump or not to jump determined by condition-code flags
  - – field in the op code indicates how this is computed
  - – in assembler language, simply say
    - » je
      - jump on equal
    - » jne
      - jump on not equal
    - » jgt
      - jump on greater than
    - » etc.

Addresses

```
int a, b, c, d;

int main() {
   a = (b + c) * d;
   ...
}
```

```
mov    b,%acc
add    c,%acc
mul    d,%acc
mov    %acc,a
```

```
mov    1004,%acc
add    1008,%acc
mul    1012,%acc
mov    %acc,1000
```



Addresses

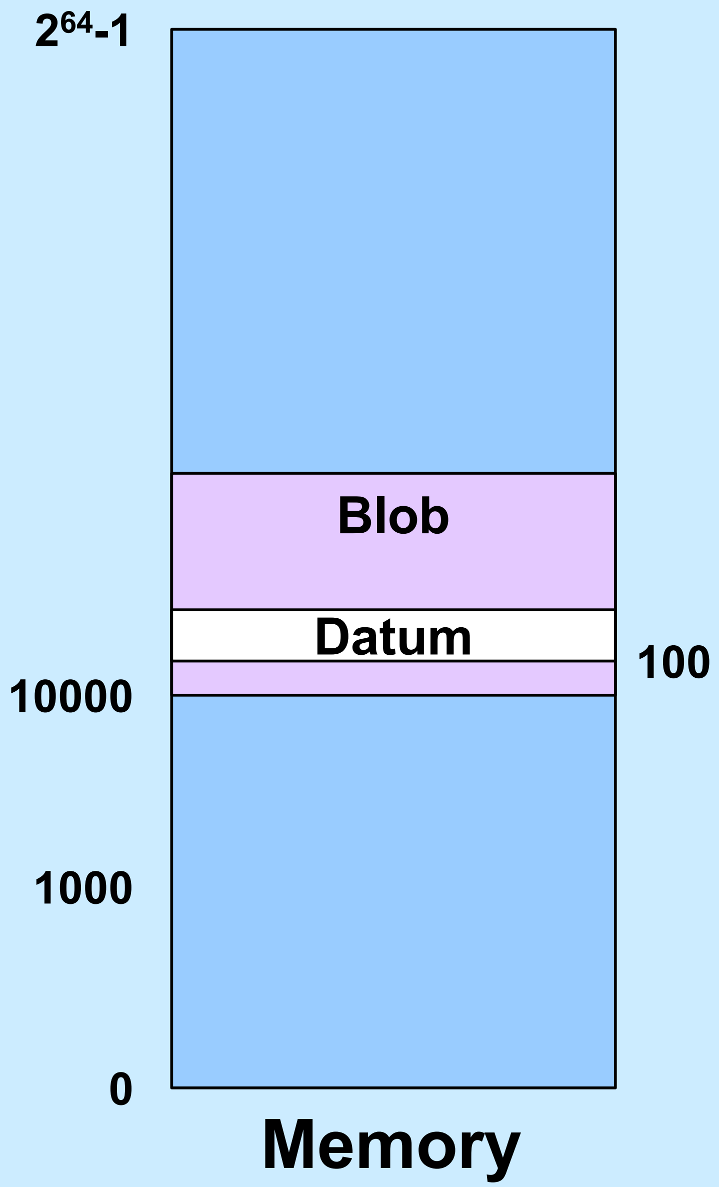
```
int b;

int func(int c, int d) {
    int a;
    a = (b + c) * d;
    ...
}

    mov    ?,%acc
    add    ?,%acc
    mul    ?,%acc
    mov    %acc,?
```

- One copy of *b* for duration of program's execution
  - *b*'s address is the same for each call to *func*
- Different copies of *a*, *c*, and *d* for each call to *func*
  - addresses are different in each call

Relative Addresses

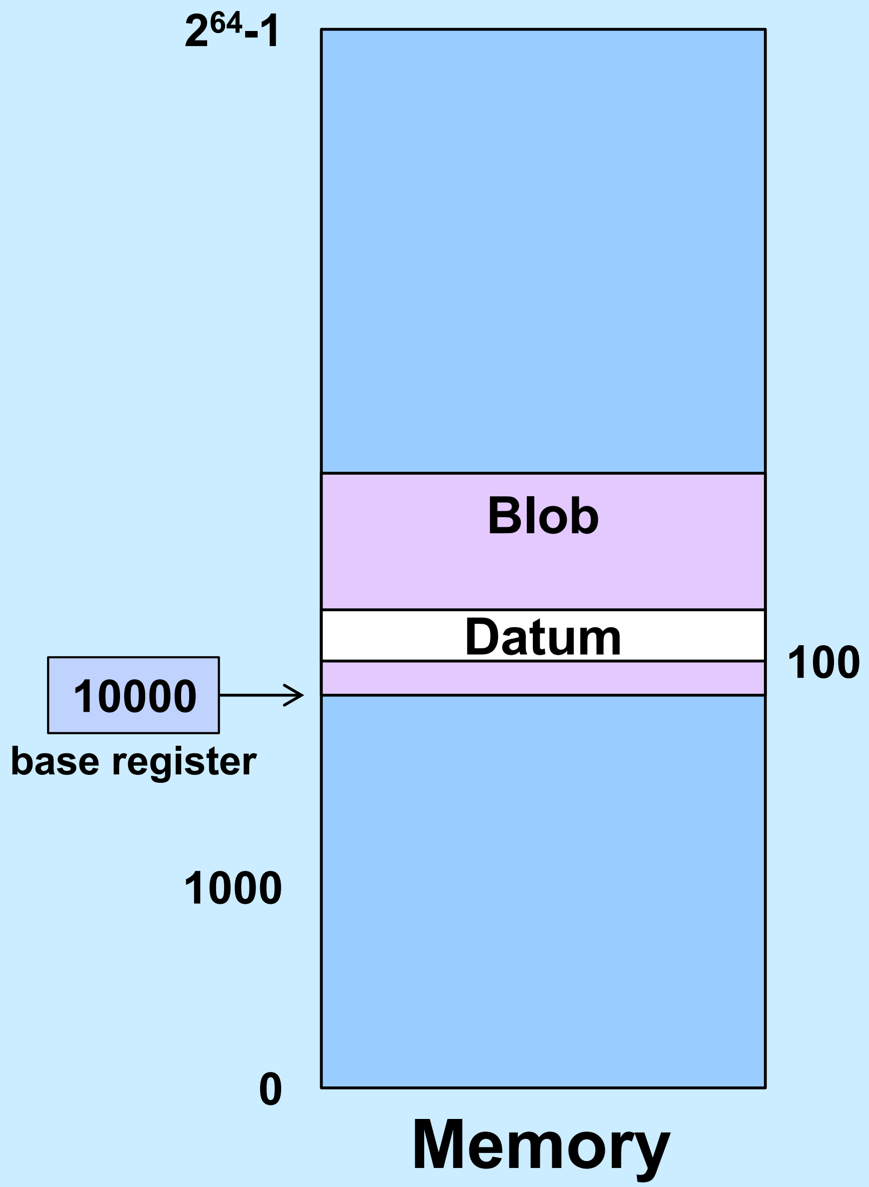
- **Absolute address**
  - actual location in memory
- **Relative address**
  - offset from some other location

- Blob's absolute address is 10000
- Datum's relative address (to Blob) is 100
  - its absolute address is 10100



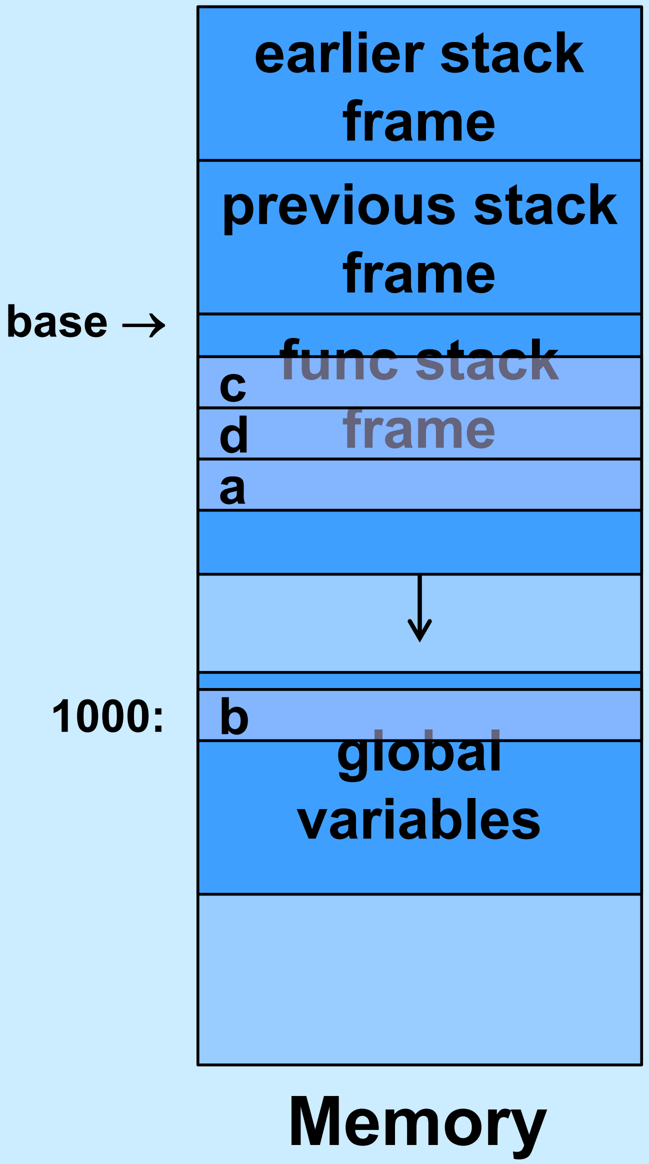
Base Registers

```
mov $10000, %base
mov $10, 100(%base)
```



Addresses

```
int b;

int func(int c, int d) {
    int a;
    a = (b + c) * d;
    ...
}

    mov    1000,%acc
    add    c_rel(%base),%acc
    mul    d_rel(%base),%acc
    mov    %acc,a_rel(%base)
```

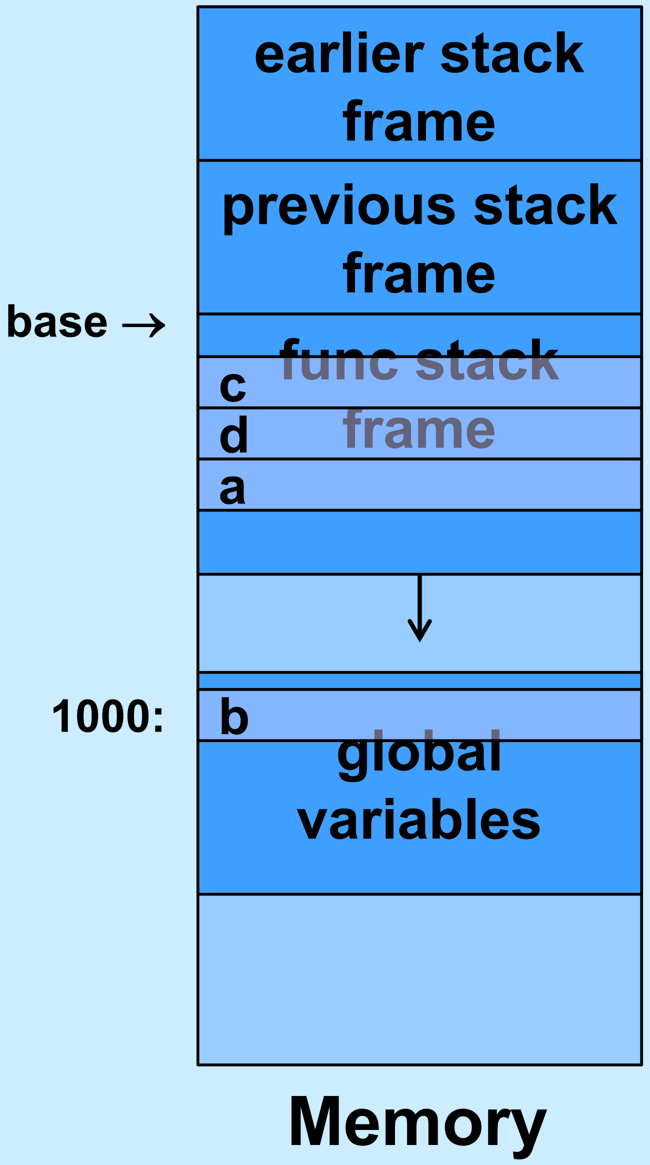


Quiz 2

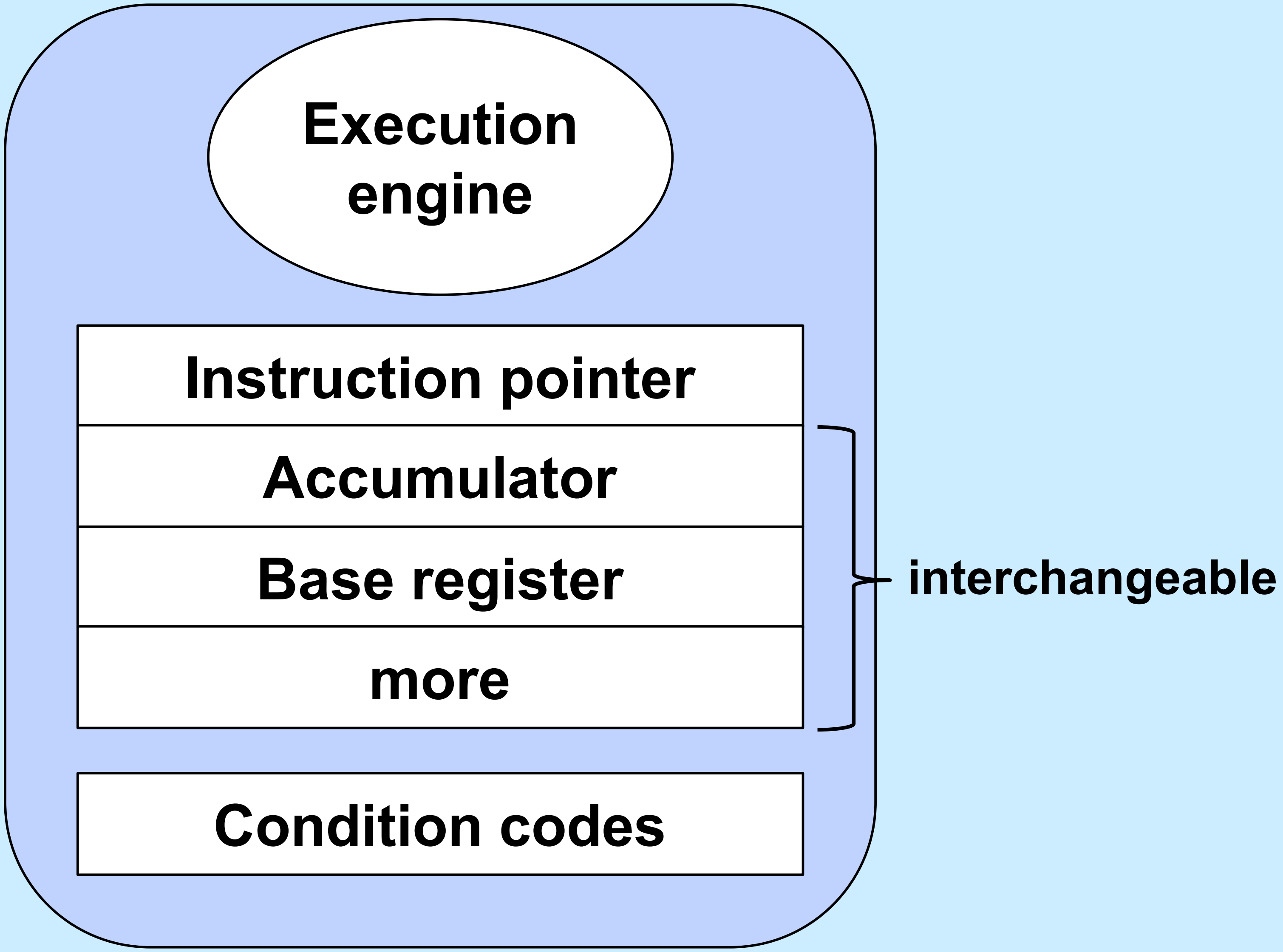
Suppose the value in *base* is 10,000 and *c_rel* is -8. What is the address of *c*?

a) 9992
b) 9996
c) 10,004
d) 10,008

```
mov    1000,%acc
add    c_rel(%base),%acc
mul    d_rel(%base),%acc
mov    %acc,a_rel(%base)
```



Registers


Execution engine
Instruction pointer
Accumulator
Base register
more
interchangeable
Condition codes