Lecture 25: Microprogramming

- The Microprogram
- Encoding instructions
- Decoding Instructions

The Microprogram

- microprogram from handout

Fetch and Decode

```
0: MAR := PC; READ; // gets low-order 13 bits
1: READ;                    // data returned in MBR
2: PC := PC + 1;
3: IR := MBR; if N then goto 25 // check bit 0 of opcode
4: TMP :=lshift(IR + IR); if N then goto 16 // check bit 1
5: TMP := TMP; if N then goto 10; // check bit 2
```

ADD

```
6: MAR := IR; READ;                 // ADD (000)
 7: READ;
8: ACC := MBR + ACC;
9: goto 0;
```

SUB

```
10: MAR := IR; READ;                    // SUB (001)
11: READ;
12: ACC := ACC + 1;
13: TMP := com(MBR);                    // 1's complement
14: ACC := ACC + TMP;
15: goto 0;
```

LOAD

```
16: TMP := TMP; if N then goto 21 // check bit 2
17: MAR := IR; READ;                    // LOAD (010)
18: READ;
19: ACC := MBR;
20: goto 0;
```

STORE

```
21: MAR := IR;                          // STORE (011)
22: MBR := ACC; WRITE;
23: WRITE;
24: goto 0;
```

More Decoding

```
25: TMP := lshift(IR+IR); if N then goto 0;
                                        // No opcodes 11...
26: TMP := TMP; if N then goto 29
```

JUMP

```
27: PC := and(IR, AMASK);       // JUMP (100)
28: goto 0;
```

JZER

```
29: ACC := ACC; if Z then goto 27;      // JZER (101)
30: goto 0;
```

```
27: PC := and(IR, AMASK);       // JUMP (100)
28: goto 0;
```

MicroInstruction Format

- review format (handout)

- Instruction encoding example:
  2: PC := PC + 1
- Fields:
  MUX =
  COND =
  ALU =
  SH =
  MBR =
  MAR =
  RD =
  WR =
  ST =
  C =
  B-Latch =
  A-Latch =
  ADDR =
- Answer:

- Another example:
  13: TMP := com(MBR)
- Fields:
  MUX =
  COND =
  ALU =
  SH =
  MBR =
  MAR =
  RD =
  WR =
  ST =
  C =
  B-Latch =
  A-Latch =
  ADDR =
- Answer:

- More than one MAL instruction per microinstruction!:
  16: TMP := TMP; if N then goto 21
- Fields:
  MUX =
  COND =
  ALU =
  SH =
  MBR =
  MAR =
  RD =
  WR =
  ST =
  C =
  B-Latch =
  A-Latch =
  ADDR =
- Answer:

- Another example:
  17: MAR := IR; READ;
- Fields:
  MUX =
  COND =
  ALU =
  SH =
  MBR =
  MAR =
  RD =
  WR =
  ST =
  C =
  B-Latch =
  A-Latch =
  ADDR =
- Answer:

- Another example:
  27: PC := and(IR, AMASK)
- Fields:
  MUX =
  COND =
  ALU =
  SH =
  MBR =
  MAR =
  RD =
  WR =
  ST =
  C =
  B-Latch =
  A-Latch =
  ADDR =
- Answer:

Decoding

- Decoding hints:
  - break the binary up into the fields as shown
  - an unused field is not necessarily going to be zero! Any non applicable fields (addresses if not jumping, B-latch values for single operand instructions) should be ignored.

- Decode Example

0 01 00 01 0 0 0 0 1 011 010 010 000000

- Fields:
  MUX =
  COND =
  ALU =
  SH =
  MBR =
  MAR =
  RD =
  WR =
  ST =
  C =
  B-Latch =
  A-Latch =
  ADDR =
- Answer:

- Decode Example

0 11 10 00 0 0 0 0 0 011 101 010 000000

Fields:
MUX =
COND =
ALU =
SH =
MBR =
MAR =
RD =
WR =
ST =
C =
B-Latch =
A-Latch =
ADDR =

- Answer: ???

In-Class Exercise

- Create the binary microinstruction for the following:
  MBR := ACC + 1; WRITE;
- Write MAL code for the following:
  0 10 10 10 0 0 0 0 1 011 001 011 001100

MBR := ACC + 1; WRITE;

- Fields:
  - MUX: 0 – we're getting our input from A-latch
  - COND: 00 – no jump
  - ALU: 00 – we're adding
  - SH: 00 – don't shift
  - MBR: 1 – we're storing our result in MBR
  - MAR: 0 – we're not storing anything in MAR
  - RD: 0 – not reading
  - WR: 1 – we're writing!
  - ST: 0 – we're not storing into a general register.
  - C: doesn't matter (storing into MBR)
  - B-latch: 101 (our second operand is 1)
  - A-latch: 000 (our operand is ACC)
  - ADDR: doesn't matter.
- Answer:

0 00 00 00 1 0 0 1 0 000 101 000 00000

0 10 10 10 0 0 0 0 1 011 001 011 001100

Fields:

- MUX: 0 – we're getting our input from A-latch
- COND: 10 – jump if Z=1
- ALU: 10 – pass through
- SH: 10 – right shift
- MBR: 0 – we're not storing anything in MBR
- MAR: 0 – we're not storing into MAR
- RD: 0 –not reading
- WR: 0 – not writing
- ST: 1 – storing into a general register
- C:011 – (3 – TMP register)
- B-latch: 001 (PC)
- A-latch: 011 (our operand is also TMP)
- ADDR: 001100

- Answer: ???

  TMP := rshift(TMP); If Z then goto 12