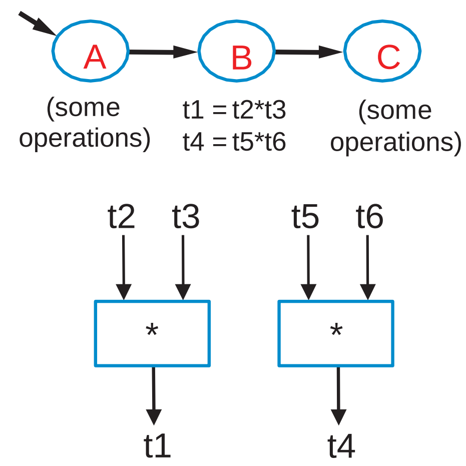
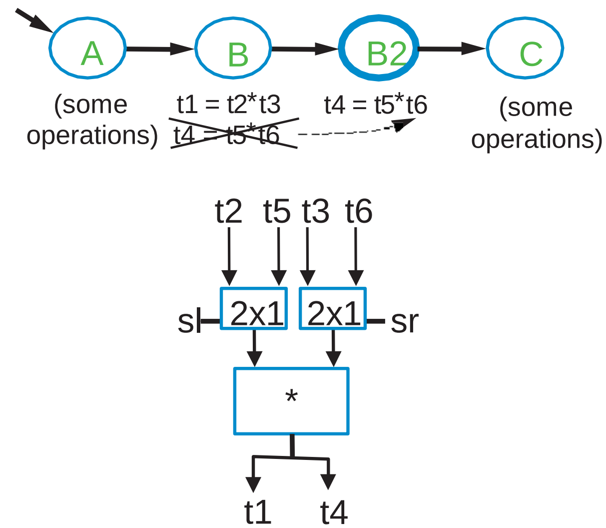
8 – High-Level Synthesis: Scheduling

ECE 474A/574A
COMPUTER-AIDED LOGIC DESIGN

Scheduling

- What have we done?
  - Specified the order in which operations are performed
- What's next - Operator Scheduling
  - Assigning operations performed in each states
  - Generally speaking, determining the start time of each task/operation
- Why is scheduling important?
  - Determines the amount of concurrency of the resulting implementation – affects performance
  - Maximum amount of concurrent operations of a given type at any time step also determines the amount of hardware resources of that type required – effects area





Control/Data flow graph (CDFG)

- How do we represent a scheduling?
- Dataflow graph – represents the way data flows through a computation
  - Computations limited to 2-input

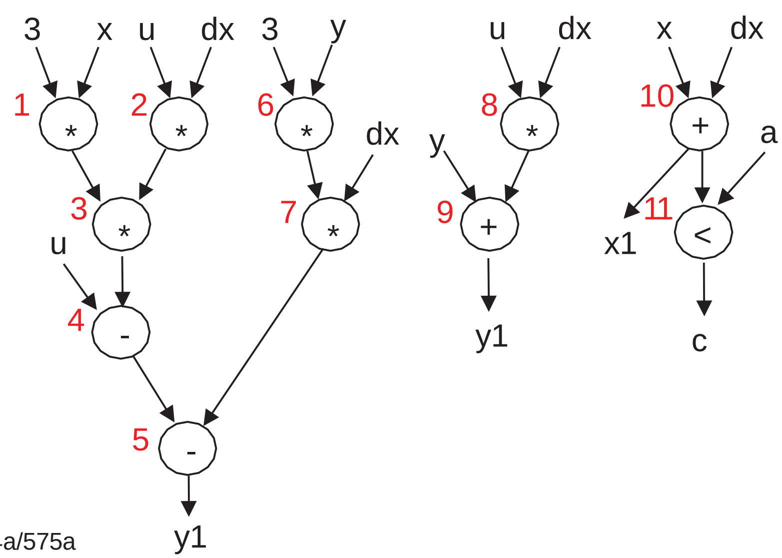
Code fragment we want to implement

x1 = x + dx;
u1 = u – (3 * x * u * dx) – (3 * y * dx)
y1 = y + u * dx
c = x1 < a

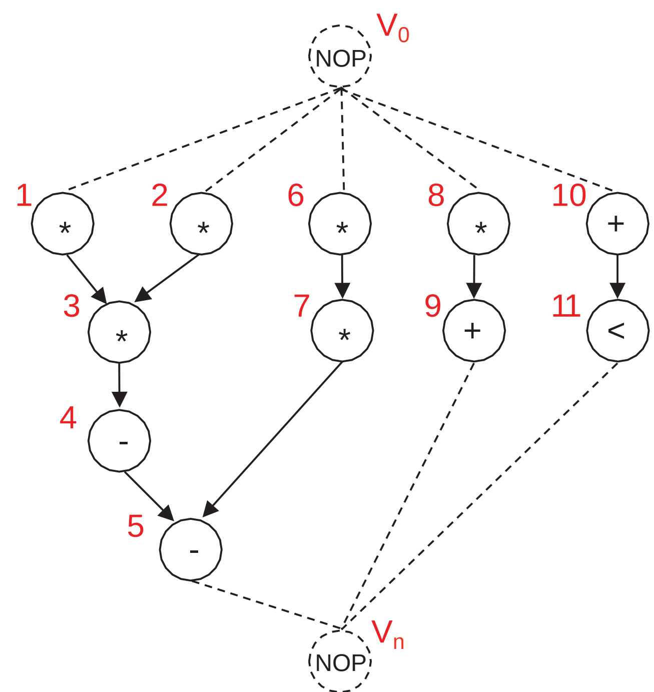
Task Representation - Sequencing graph

- Determining a schedule
  - Don't really care about the actual input/output values
  - Just want to know the task we perform (add, subtract, compare, etc..) and the dependencies among the task
- Utilize a Sequencing graph

CDFG



Sequencing Graph



Sequencing graph

Source node

Represented by a NOP (no operation) node

Indicates start of computation

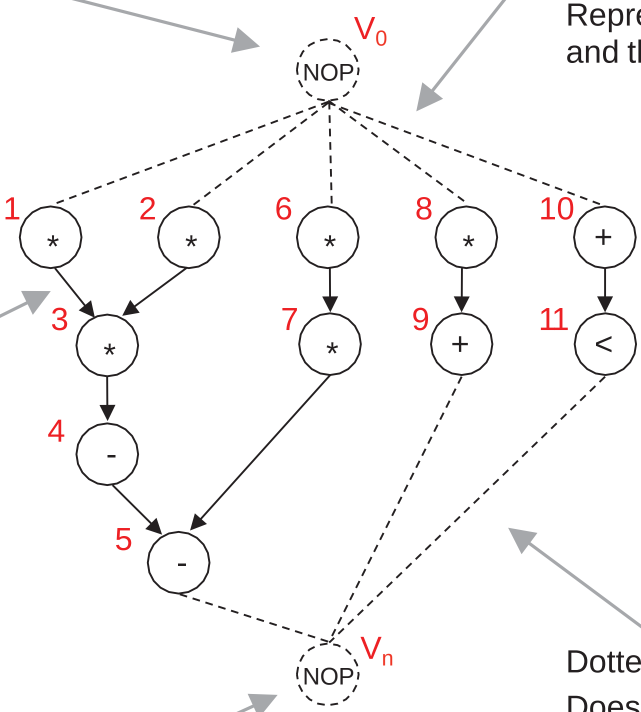
Dotted lines

Does NOT represent a dependency between tasks or operations

Represents a connection between the source and the initial task or operations

Vertices

Represents the task or operation to be performed



Edges

Represents dependencies among operations

Cannot perform operation 3 until operation 1 and 2 are completed

Dotted lines

Does NOT represent a dependency between tasks or operations

Represents a connection between the sink and the final task or operations

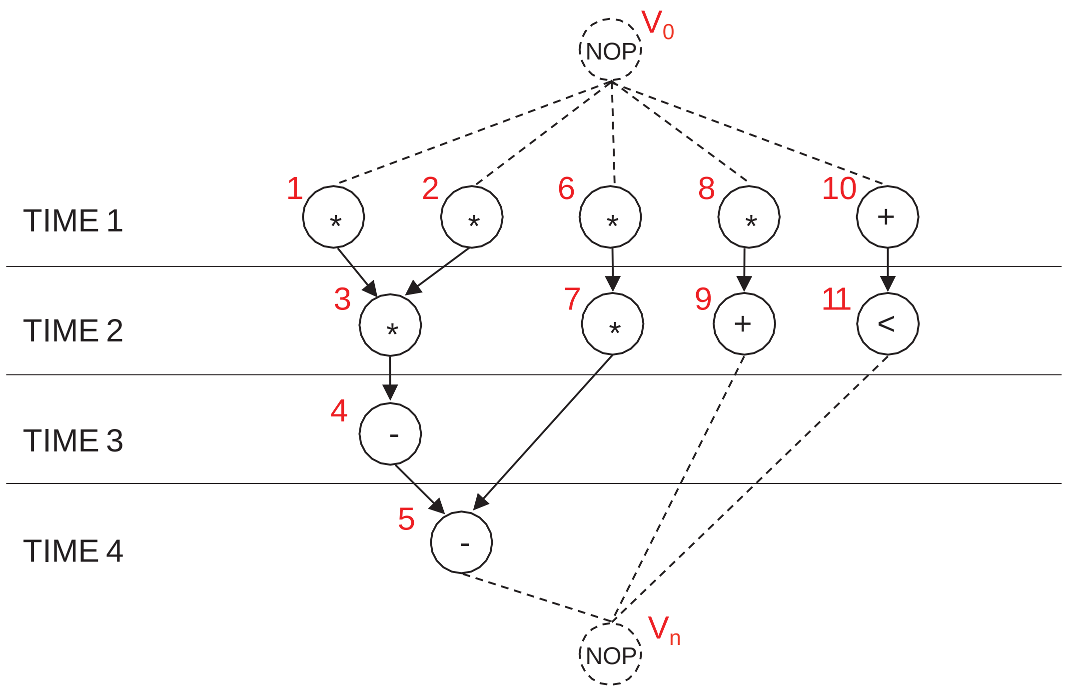
Sink node

Represented by a NOP (no operation) node

Indicates completion of computation

Scheduling - Sequencing Graphs

- Sequencing graph itself only specifies the dependencies among tasks
- Scheduling requires we associate a start time for each task/operation in the sequencing graph
  - Introduce concept of time



Operation 1, 2, 6, 8, and 10 are performed in the first clock cycle

Operation 3, 7, 9, and 11 are performed in the second clock cycle

Operation 4 is performed in the third clock cycle

Operation 5 is performed in the fourth clock cycle

Notice all operations require only 1 clock cycle to execute

ASAP Scheduling

- Unconstrained minimum-latency scheduling problem
  - We have infinite resources, all we want is the minimum time to perform the computation
  - Commonly referred to as ASAP (as soon as possible) scheduling

Perform ASAP scheduling on the sequencing graph

ASAP($G_S(V,E)$){
    Schedule v_0 by setting $t_0 = 1$
    repeat{
        Select a vertex v_i whose predecessors are all scheduled;
        Schedule v_i by setting $t_i = \max_{j:(v_j, v_i) \in E} t_j + d_j$
    }
    until (v_n is scheduled);
    return t;
}

Schedule the source node v_0 for time 1

Look for tasks/operations that are not dependent on a task/operation that hasn’t been scheduled yet

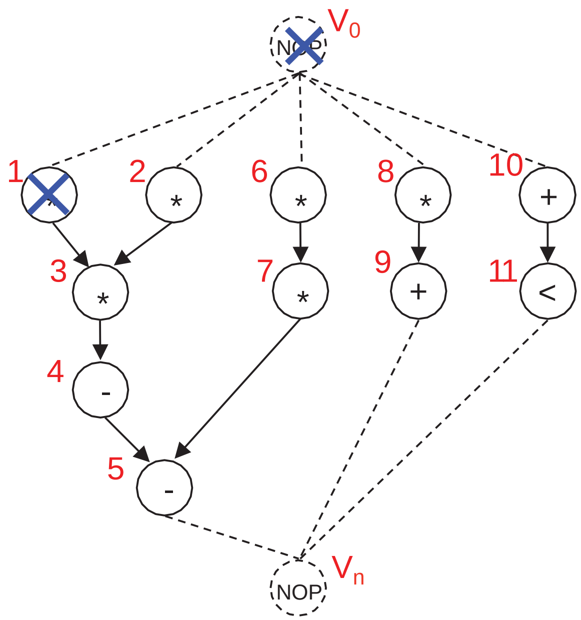
Schedule the task/operation to time = time predecessor scheduled for + time required for predecessor to execute

may have multiple predecessors, take maximum time

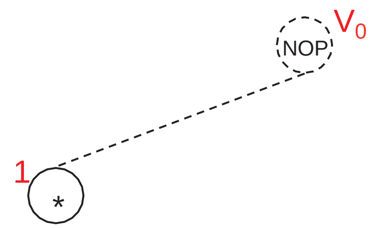
Keep going until we have scheduled the sink node v_n

ASAP Scheduling

Example 1



TIME 1



<u>Step1</u>

Schedule v_0 at time 1

<u>Step2</u>

Select a vertex v_i whose predecessors are all scheduled

All of v_1 predecessors are scheduled

<u>Step3</u>

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_0 start time + v_0 execution time
= 1 + 0
= 1

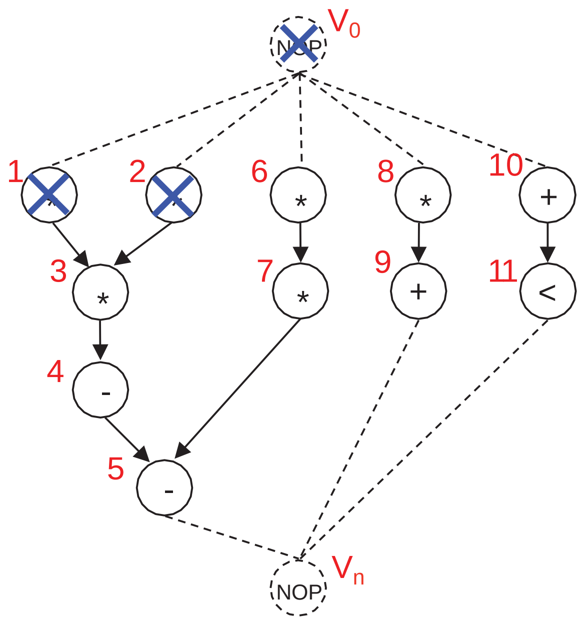
<u>Step4</u>

Has v_n been scheduled yet?

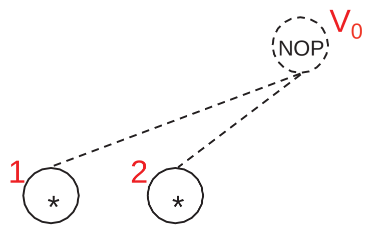
No. Repeat loop.

ASAP Scheduling

Example 1



TIME 1



Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_2 predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_0 start time + v_0 execution time
= 1 + 0
= 1

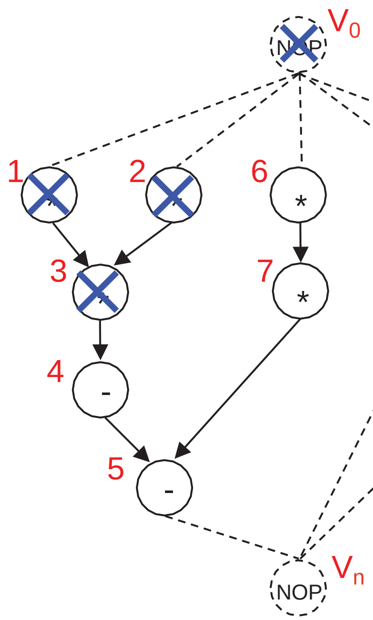
Step4

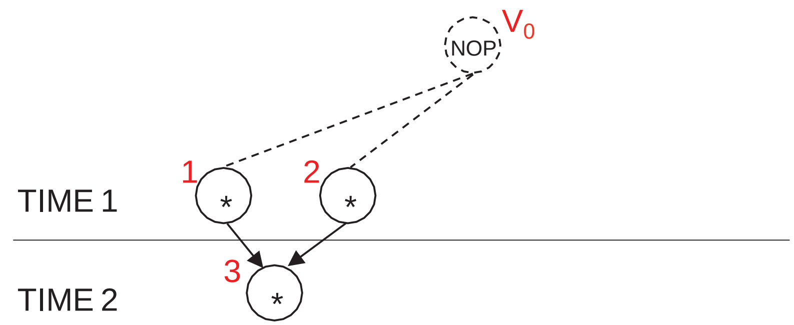
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_3 predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_1 start time + v_1 execution time
= 1 + 1
= 2

Time = v_2 start time + v_2 execution time
= 1 + 1
= 2

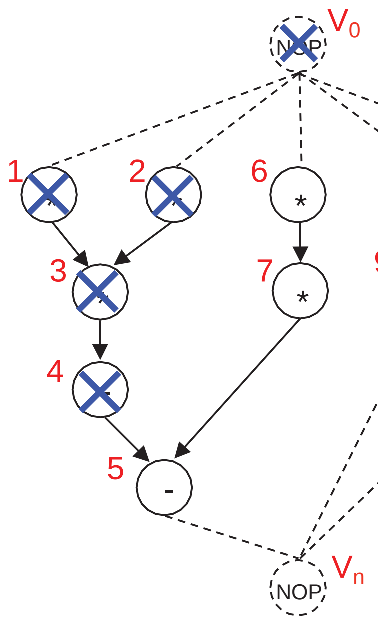
Step4

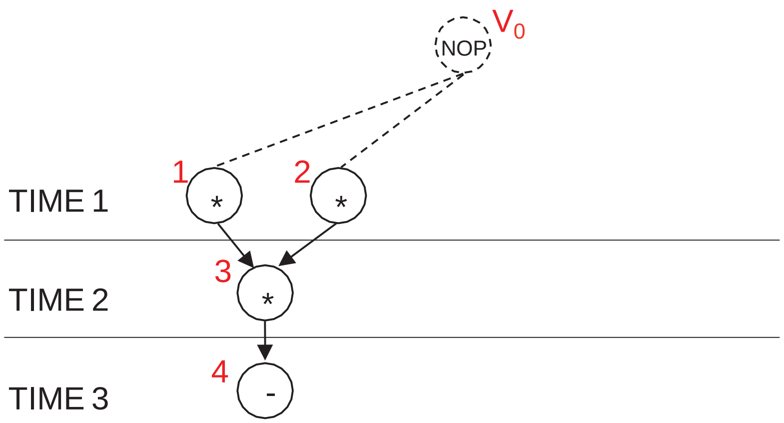
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_4 predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_3 start time + v_3 execution time
= 2 + 1
= 3

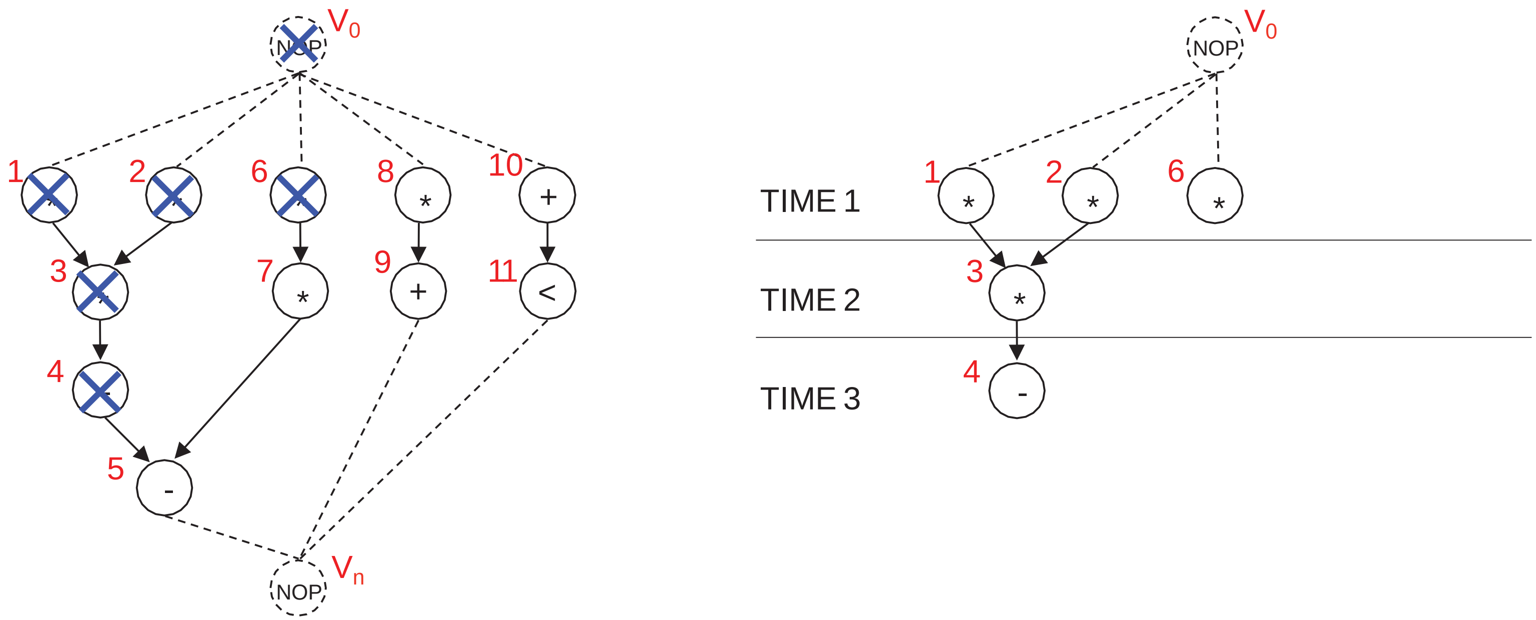
Step4

Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1



Step2

Select a vertex v_i whose predecessors are all scheduled

v_5 still has predecessors not scheduled (v_7), skip for now

All of v_6 predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_0 start time + v_0 execution time

= 1 + 0

= 1

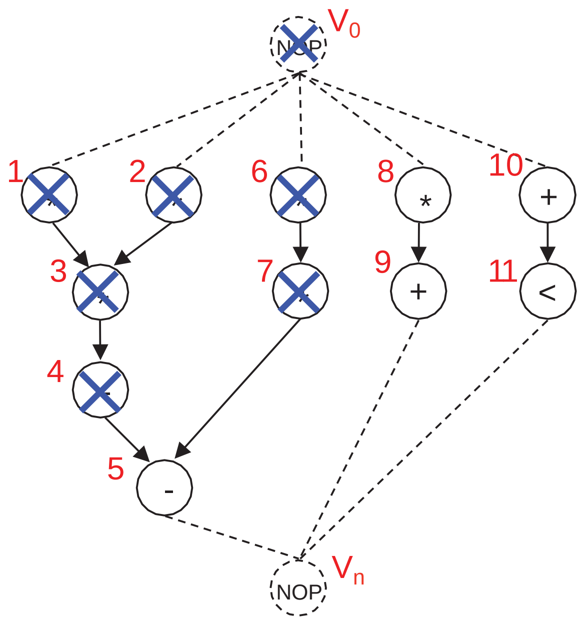
Step4

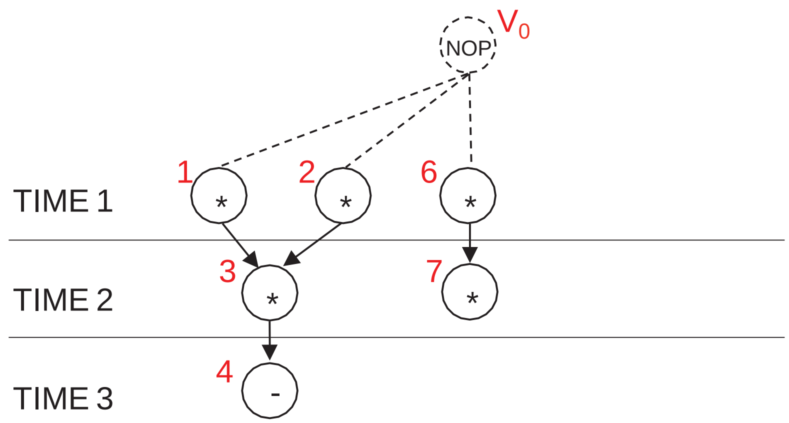
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_7 predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_6 start time + v_6 execution time
= 1 + 1
= 2

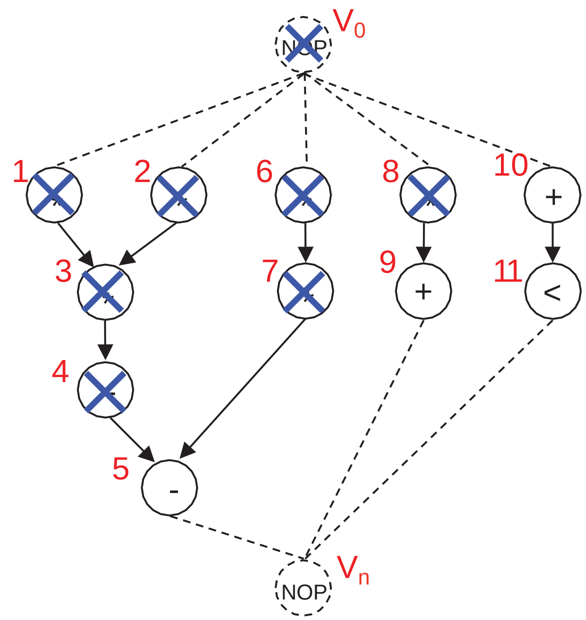
Step4

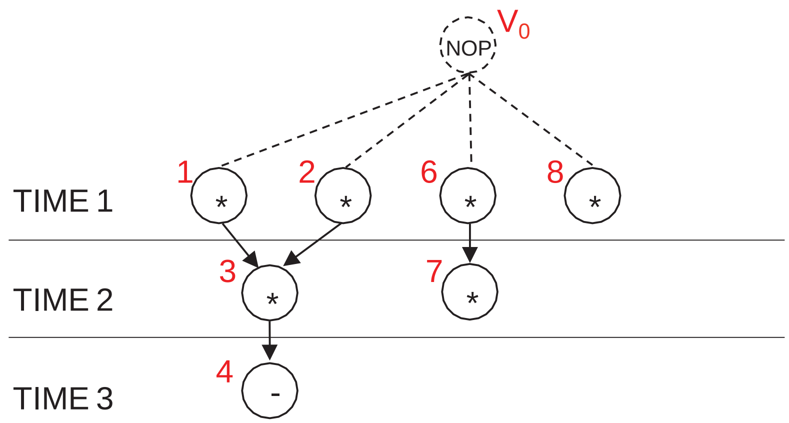
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_8 predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_0 start time + v_0 execution time
= 1 + 0
= 1

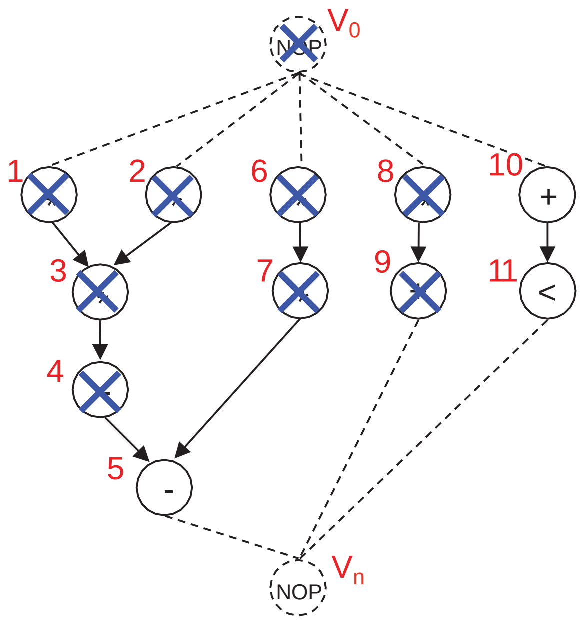
Step4

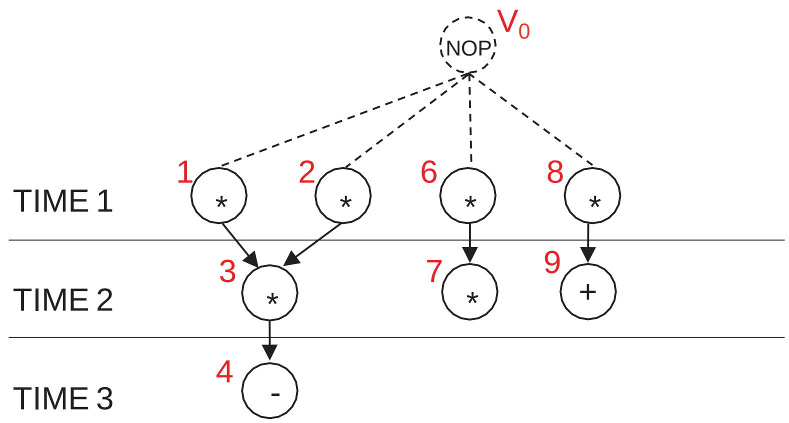
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





<u>Step2</u>

Select a vertex v_i whose predecessors are all scheduled

All of v_9 predecessors are scheduled

<u>Step3</u>

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_8 start time + v_8 execution time
= 1 + 1
= 2

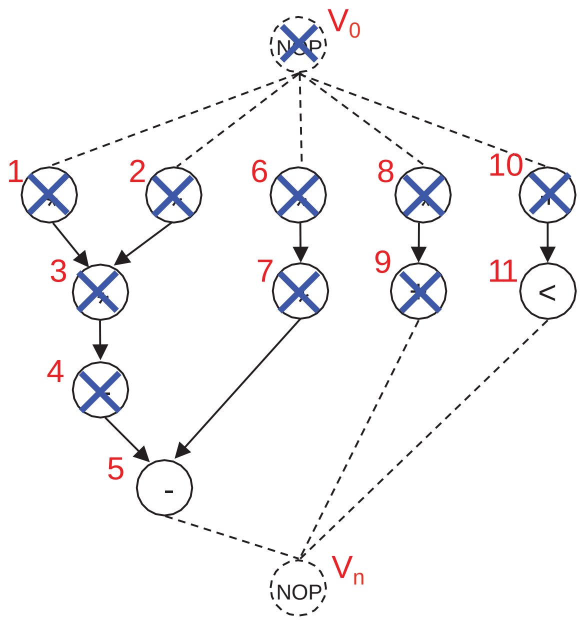
<u>Step4</u>

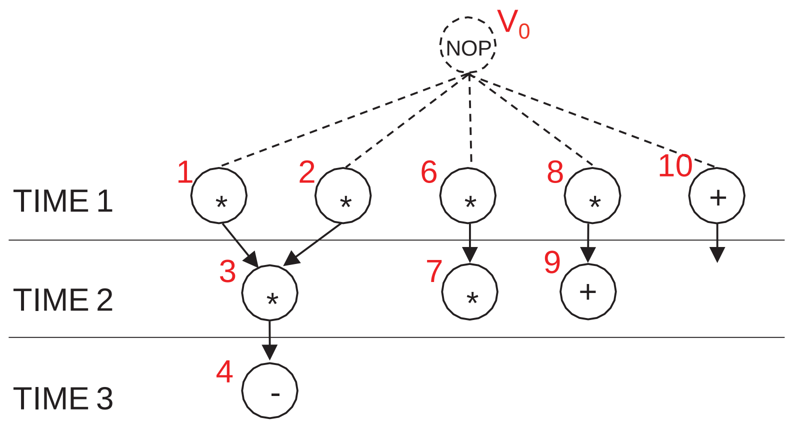
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_{10} predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_0 start time + v_0 execution time
= 1 + 0
= 1

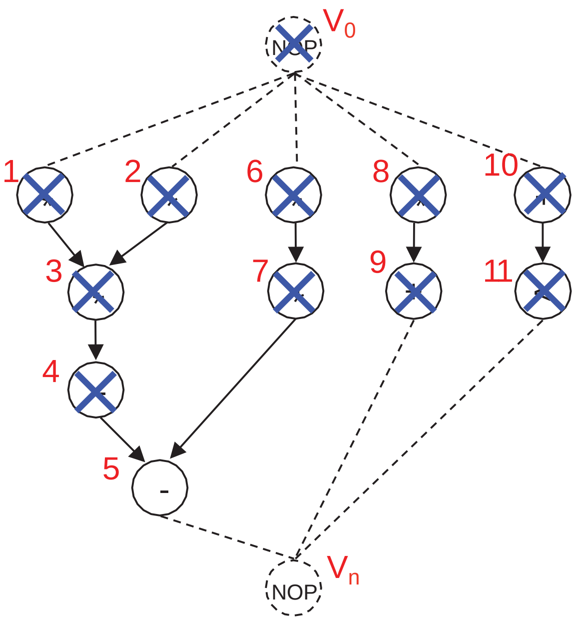
Step4

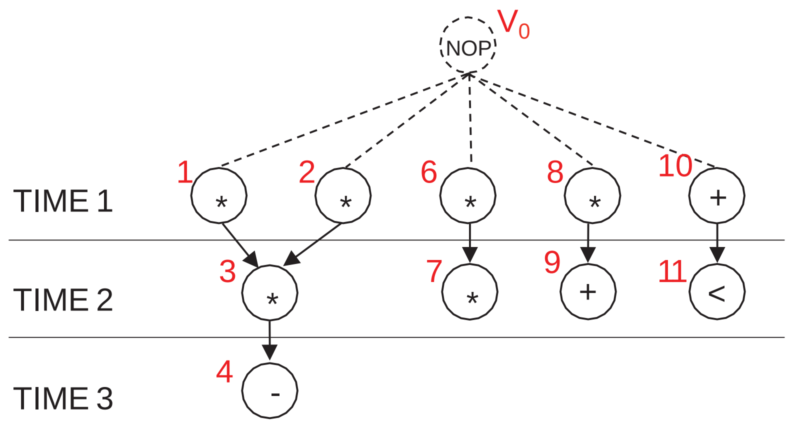
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

All of v_{11} predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_{10} start time + v_{10} execution time
= 1 + 1
= 2

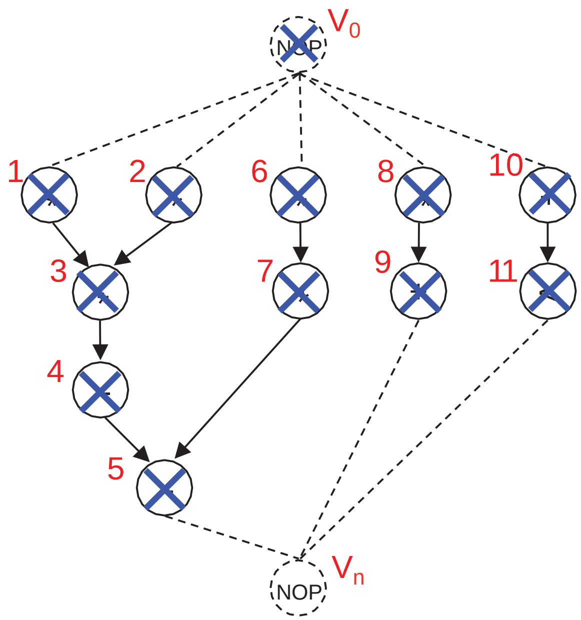
Step4

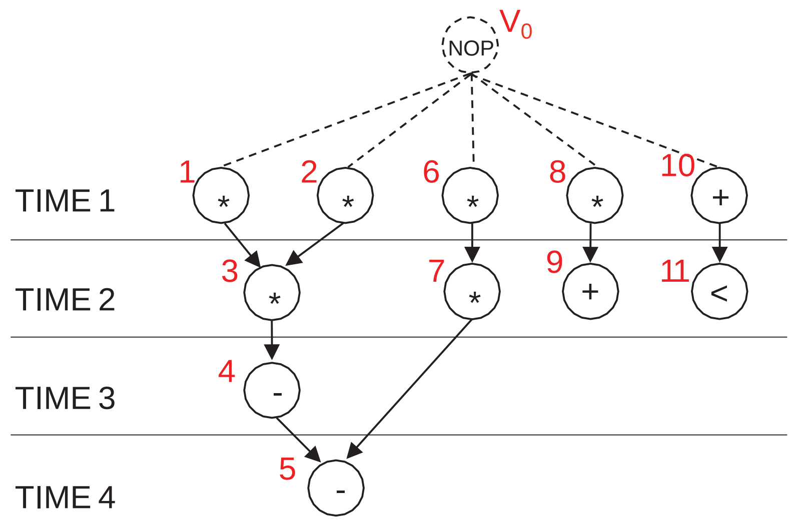
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

v_n still has predecessors not scheduled (v_5), skip for now
Return to v_5, all predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_4 start time + v_4 execution time
= 3 + 1
= 4

Time = v_7 start time + v_7 execution time
= 2 + 1
= 3

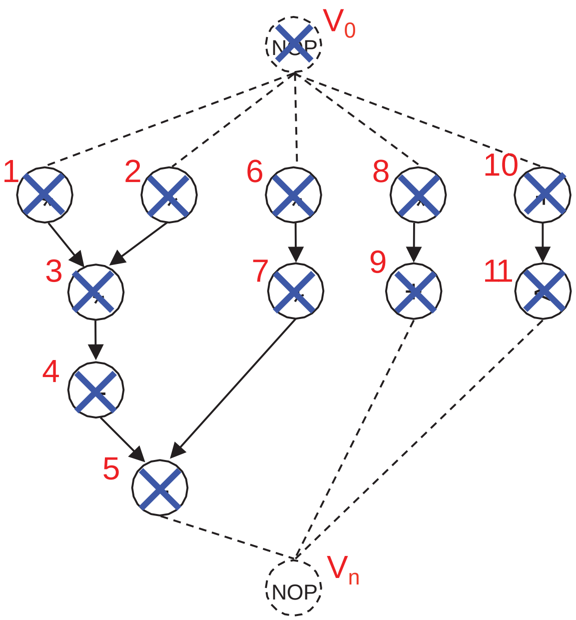
Step4

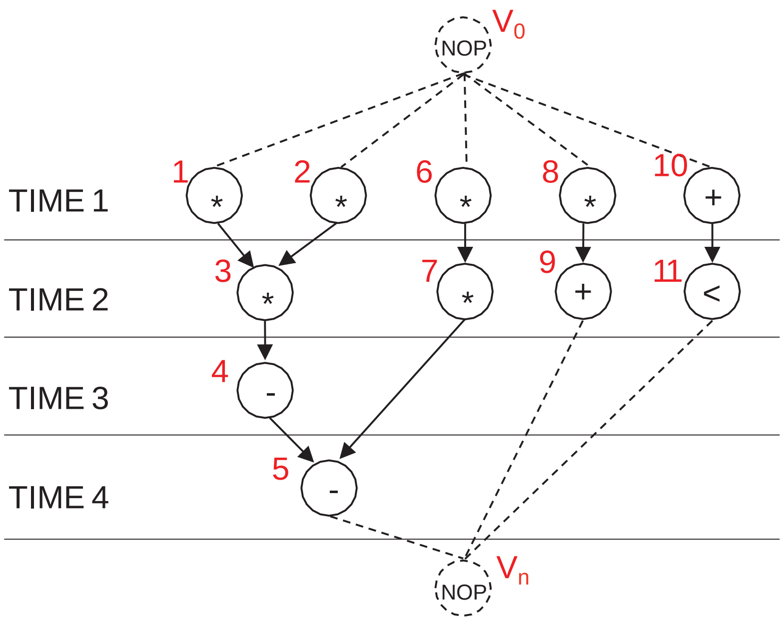
Has v_n been scheduled yet?

No. Repeat loop.

ASAP Scheduling

Example 1





Step2

Select a vertex v_i whose predecessors are all scheduled

Return to v_n, all predecessors are scheduled

Step3

Schedule v_i to time = predecessor's scheduled time + time required for predecessor to execute

Time = v_5 start time + v_5 execution time
= 4 + 1
= 5

Time = v_9 start time + v_9 execution time
= 2 + 1
= 3

Time = v_{11} start time + v_{11} execution time
= 2 + 1
= 3

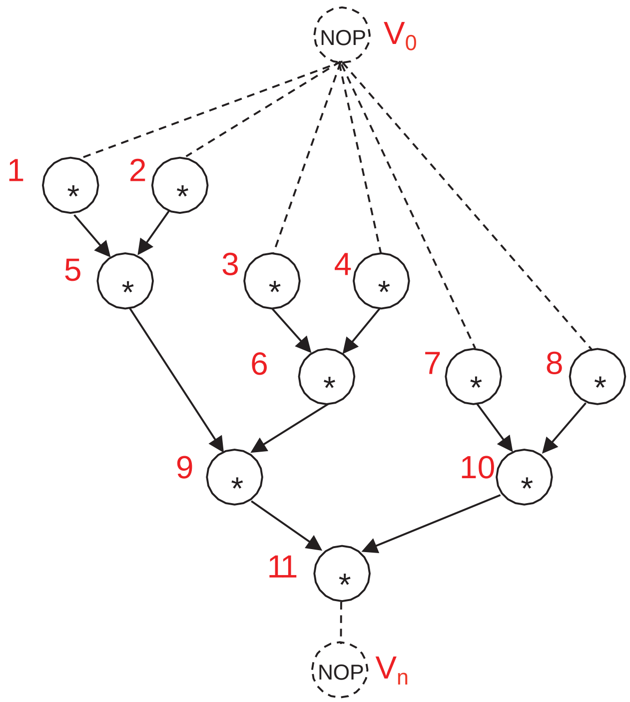
Step4

Has v_n been scheduled yet?

Yes. We are done!

ASAP Scheduling

Example 2



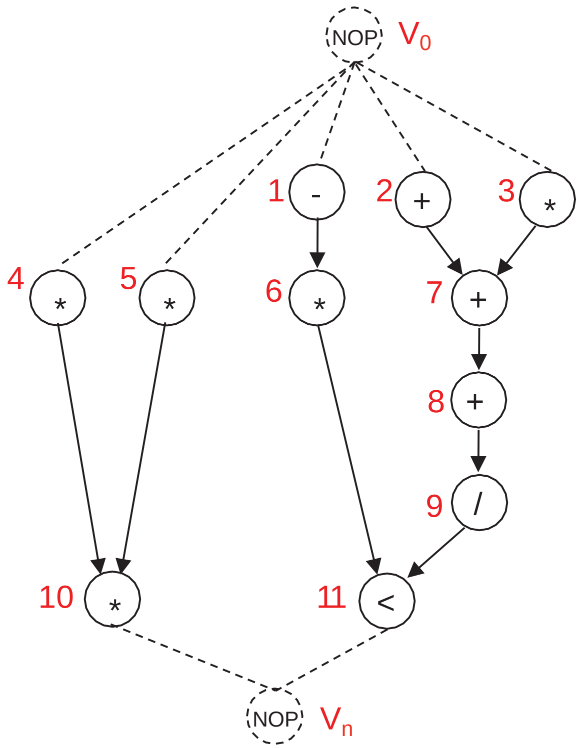
TIME 1	
TIME 2	
TIME 3	
TIME 4	

ASAP Scheduling goal is to schedule tasks/operations to perform as soon as possible

We can skip the algorithm and visually move vertices ″up″ as far as possible

ASAP Scheduling

Example 3



TIME 1

TIME 2

TIME 3

TIME 4

TIME 5

ALAP Scheduling

- Latency constrained scheduling problem
  - Schedule must satisfy an upper bound on latency
  - Commonly referred to as ALAP (as late as possible) scheduling

ALAP($G_S(V,E)$, λ){
  Schedule v_n by setting $t_n = \lambda + 1$
  repeat{
    Select a vertex v_i whose successors are all scheduled;
    Schedule v_i by setting $t_i = \min_{j:(v_j, v_i) \in E} t_j - d_j$
  }
  until (v_0 is scheduled);
  return t;
}

Perform ALAP scheduling on the sequencing graph, λ is the upper time bound

Schedule the sink node v_n for upper latency bound + 1

Look for tasks/operations whose successors are already scheduled

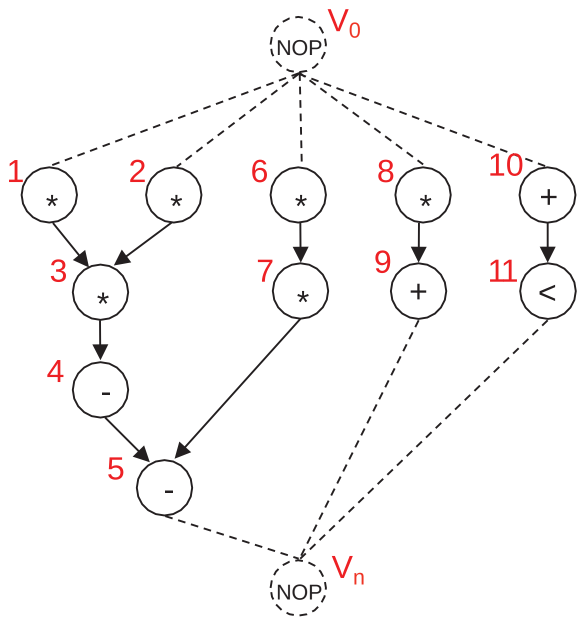
Schedule the task/operation to time = time successor scheduled for - time required for successor to execute

may have multiple successors, take minimum time

Keep going until we have scheduled the source node v_0

ALAP Scheduling

Example 1



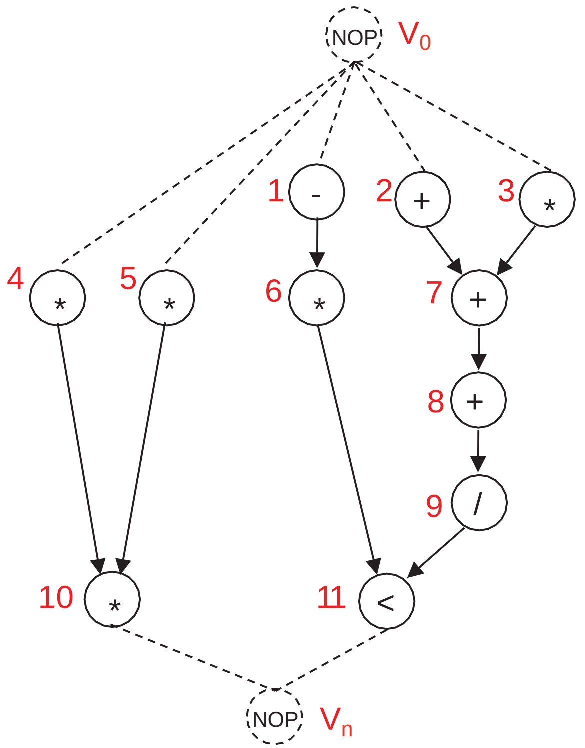
TIME 1
TIME 2
TIME 3
TIME 4

ALAP Scheduling goal is to schedule tasks/operations to perform as late as possible

We can skip the algorithm and visually move vertices ″down″ as far as possible

ALAP Scheduling

Example 3



TIME 1

TIME 2

TIME 3

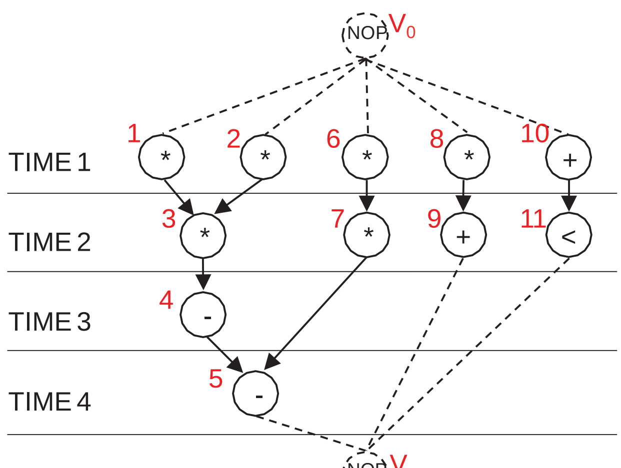
TIME 4

TIME 5

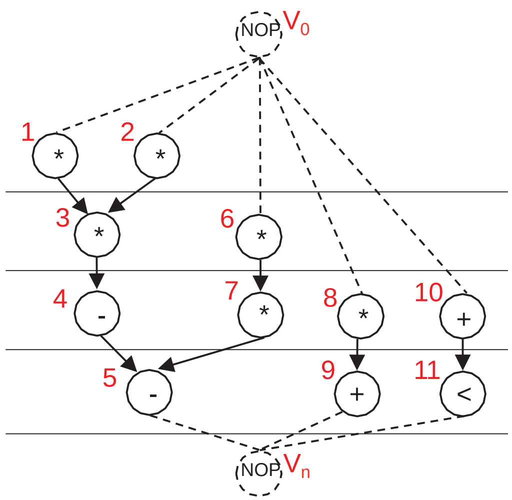
Mobility

- Mobility (or slack) important quantity used by some scheduling algorithms
  - Mobility = start time $_{\text{ALAP scheduling}}$ – start time $_{\text{ASAP scheduling}}$
  - Mobility = 0, task/operation can only be started at the given time in order to meet overall latency constraint
  - Mobility > 0, indicates span of possible start times
    - Helps with minimizing resources (adders, multipliers, etc.)

Example 1 – ASAP Schedule



Example 1 – ALAP Schedule



$$V_1 \text{ mobility} = \text{time}_{ALAP}(V_1) - \text{time}_{ASAP}(V_1) = 1 - 1 = 0$$

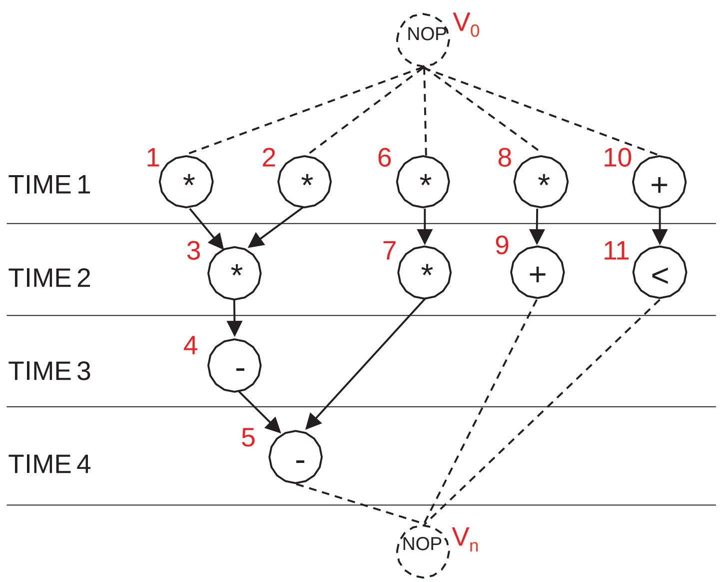
$$V_6 \text{ mobility} = \text{time}_{ALAP}(V_6) - \text{time}_{ASAP}(V_6) = 2 - 1 = 1$$

$$V_{11} \text{ mobility} = \text{time}_{ALAP}(V_{11}) - \text{time}_{ASAP}(V_{11}) = 4 - 2 = 2$$

Mobility

Example 1 – ASAP Only

- What do we get with the ASAP Schedule?
  - Latency = 4
  - Resource requirement = 4 multipliers, 2 ALUs



4 multiply operations, 1 ALU operation

2 multiply operations, 2 ALU operations

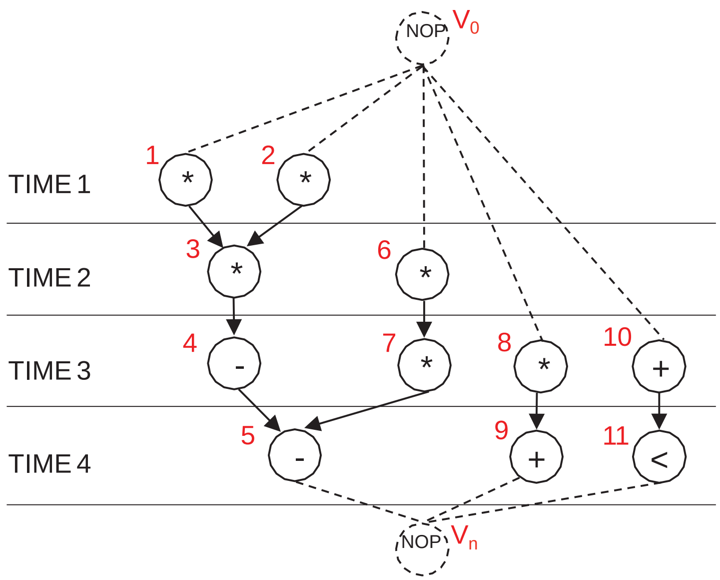
1 ALU operation

1 ALU operation

Mobility

Example 1 – ALAP Only

- What do we get with the ALAP Schedule?
  - Latency = 4
  - Resource requirement = 2 multipliers, 3 ALUs



2 multiply operations

2 multiply operations

2 multiply operations, 2 ALU operations

3 ALU operations

Mobility

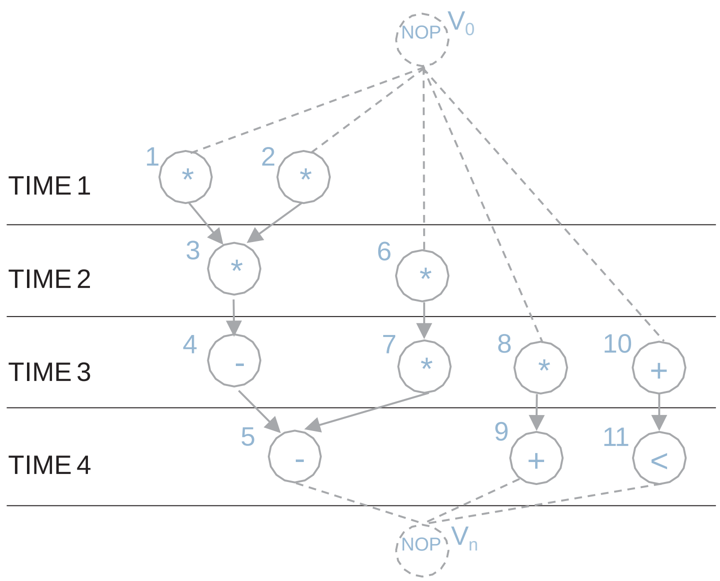
Example 1 – Modify ALAP

- Start with ALAP schedule
- Use mobility to try to improve resource requirements

Operations with mobility = 0
v_1, v_2, v_3, v_4, v_5

Operations with mobility = 1
v_6, v_7

Operations with mobility = 2
v_8, v_9, v_{10}, v_{11}



2 multiply operations

2 multiply operations

2 multiply operations, 2 ALU operations

3 ALU operations

Mobility

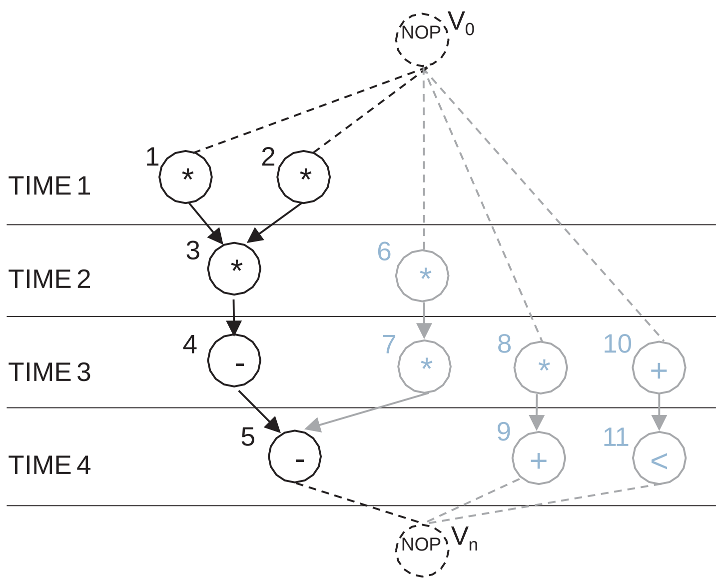
Example 1 – Modify ALAP

- Start with ALAP schedule
- Use mobility to try to improve resource requirements
  - Vertices with mobility = 0 cannot be moved, they are part of the critical path

Operations with mobility = 0
v_1, v_2, v_3, v_4, v_5

Operations with mobility = 1
v_6, v_7

Operations with mobility = 2
v_8, v_9, v_{10}, v_{11}



2 multiply operations

2 multiply operations

2 multiply operations, 2 ALU operations

3 ALU operations

Mobility

Example 1 – Modify ALAP

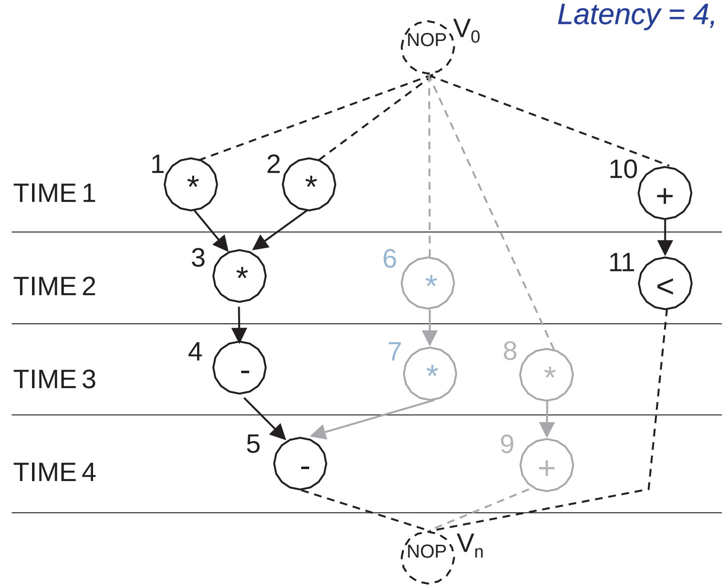
- Start with ALAP schedule
- Use mobility to try to improve resource requirements
  - Vertices with mobility = 0 cannot be moved, they are part of the critical path
  - Vertices with mobility > 0 can be moved to minimize resource requirements

Latency = 4, Resources = 2 multipliers, 2 ALUs
Latency = 4, Resources = 4 multipliers, 2 ALUs (ASAP)
Latency = 4, Resources = 2 multipliers, 3 ALUs (ALAP)

Operations with mobility = 0
v_1, v_2, v_3, v_4, v_5

Operations with mobility = 1
v_6, v_7

Operations with mobility = 2
v_8, v_9, v_{10}, v_{11}



2 multiply operations, 1 ALU operations

2 multiply operations, 1 ALU operations

2 multiply operations, ~~2~~ 1 ALU operations

~~3~~ 2 ALU operations

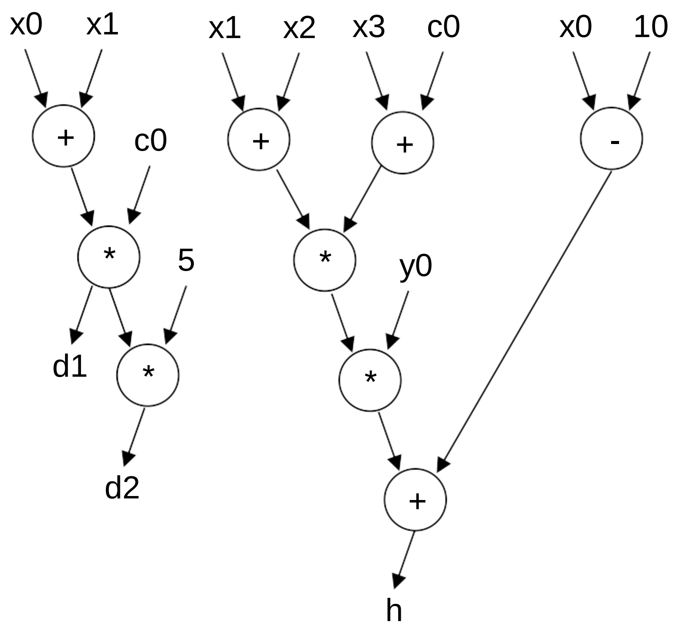
Exercise

- Convert the following C code segment to a control dataflow graph (CDFG)

```
d1 = (x0 +x1) *  c0
d2 = d1 * 5;
e = (x1 + x2) * (x3 + c0)
f = e * y0
g = x0 –10
h = f + g
```

Exercise

- From the control dataflow graph (CDFG), create the corresponding sequencing graph.



Exercise

- What is the difference in resource usage between an ASAP schedule and an ALAP schedule with a latency constraint of 7 cycles?

ASAP:

Cycle 1

Cycle 2

Cycle 3

Cycle 4

ALAP:

Cycle 1

Cycle 2

Cycle 3

Cycle 4

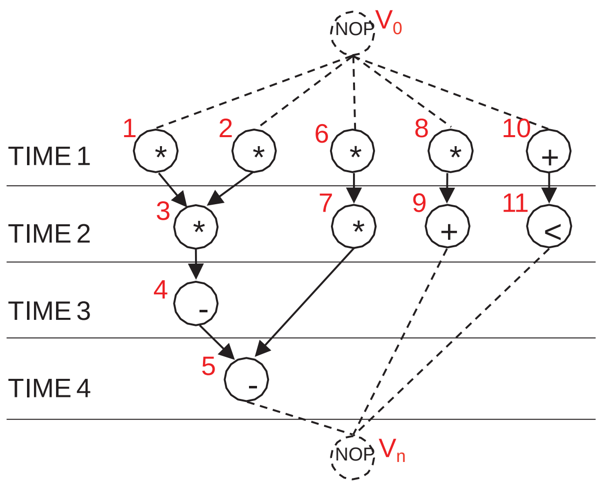
Cycle 5

Cycle 6

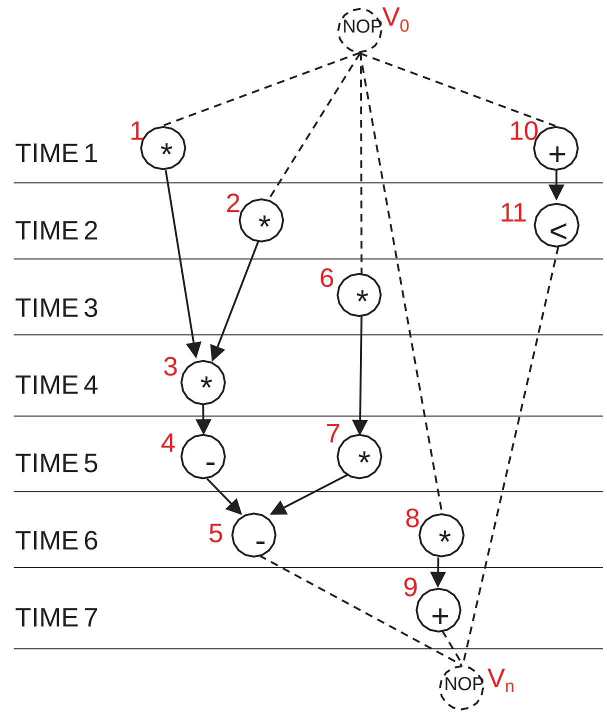
Cycle 7

Resource Constrained Scheduling

- Resource constrained scheduling problem
  - Resource usage determines circuit area
  - Consider area/latency tradeoff



ASAP schedule determines the minimum latency, we assumed infinite resources



We can determine a schedule to consider only minimizing resources – assuming latency doesn’t matter

Likely we want something in between

Hu's Algorithm

- Exact (polynomial-time) algorithm for resource constrained scheduling
  - Assumes one resource handles all possible operations
  - Assumes all operations have 1 unit delay

```
HU( G_S(V,E), a ){
    Label the vertices;
    l = 1;
    repeat {
      U = unscheduled vertices in V without predecessors
            or whose predecessors have been scheduled;
      Select S ⊂ U vertices, such that |S| ≤ a and labels in S are maximal;
      Schedule the S operations at step l by setting t = l   i : ∀_i є S;
      l = l + 1;
    } until (v_n is scheduled);
    return t;
}
```

Value of a indicates the number of resources we have available

Label with distance of vertices to sink node

Indicates the time step

Make a list of all vertices not waiting on another operation to be scheduled

Select a subset of the vertices in U, no more than a, choosing vertices with largest labels

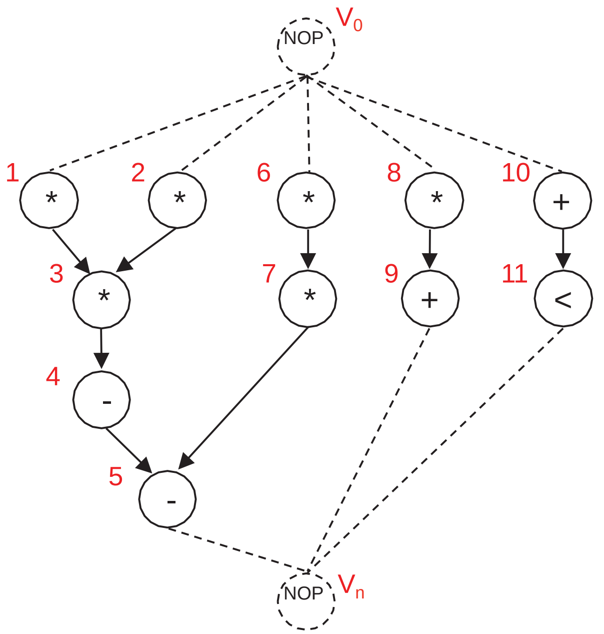
Schedule the vertices in the subset to start at time l

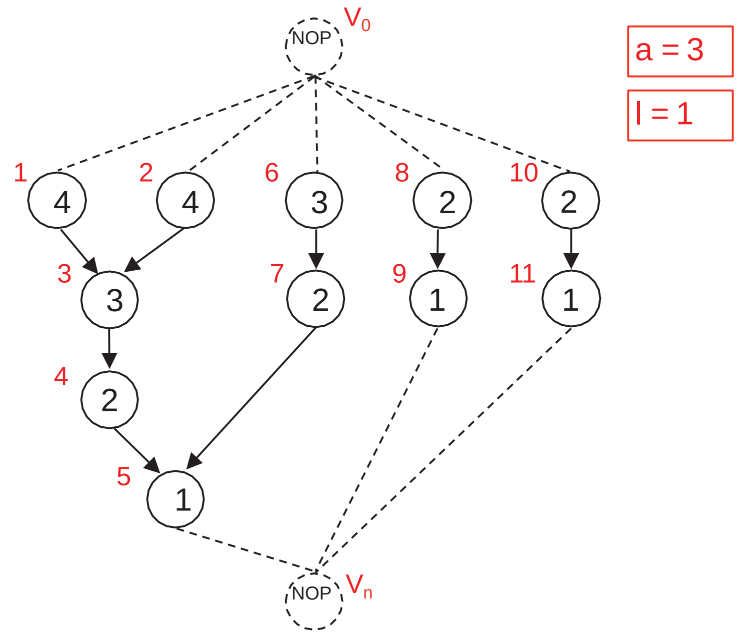
update l to next time step

Keep going until we have scheduled the sink node v_n

Hu's Algorithm

Example 1



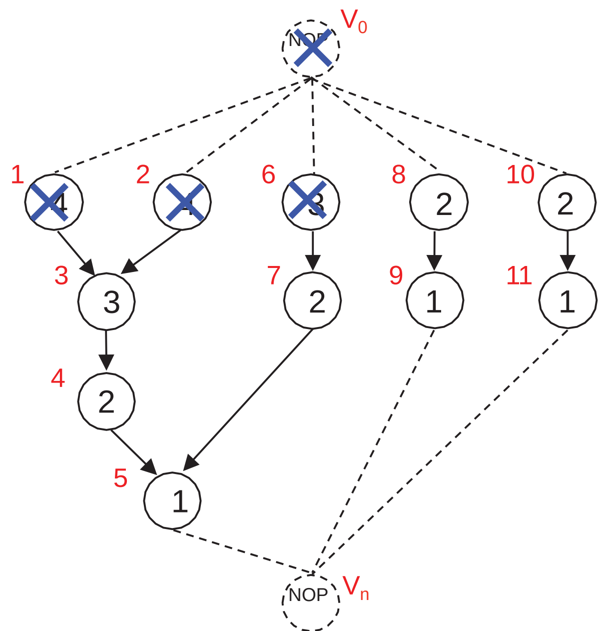


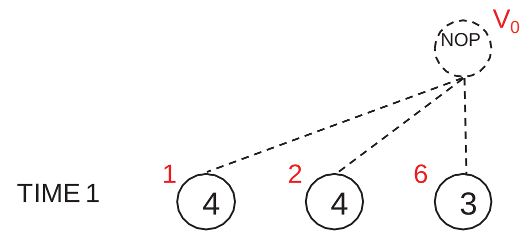
<u>Step1</u>
Label all vertices with distance to sink

<u>Step2</u>
l = 1

Hu's Algorithm

Example 1





a = 3

l = 1

<u>Step 3</u>
U = unscheduled vertices in V without predecessors or whose predecessors have been scheduled

U = { v_1, v_2, v_6, v_8, v_{10} }

<u>Step 4</u>
S = subset set of vertices in U, no more than a, where labels are maximal

S = { v_1, v_2, v_6 }

<u>Step 5</u>
Schedule vertices in S to time step l

Set vertices in S to start at 1

<u>Step 6</u>
l = l + 1

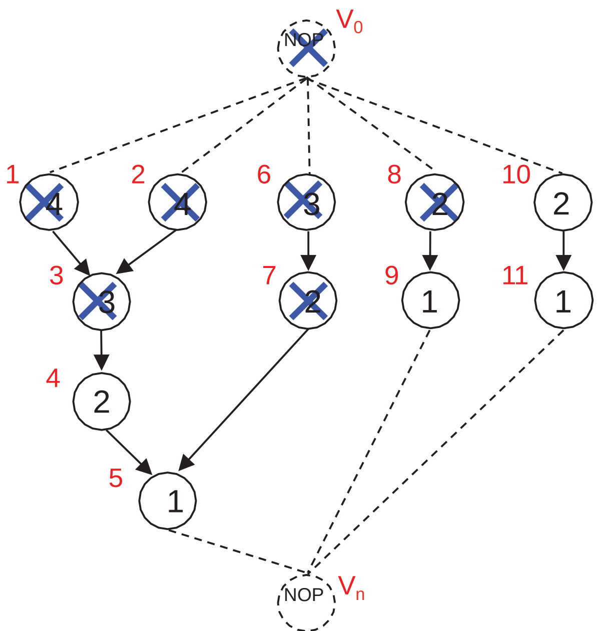
l = 1 + 1 = 2

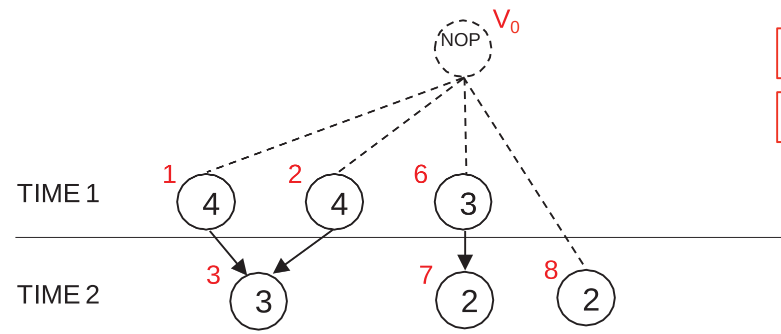
<u>Step 7</u>
Have all vertices been scheduled yet?

No. Repeat loop.

Hu's Algorithm

Example 1





a = 3

l = 2

Step 3
U = unscheduled vertices in V without predecessors or whose predecessors have been scheduled

$U = \{ v_3, v_7, v_8, v_{10} \}$

Step 4
S = subset set of vertices in U, no more than a, where labels are maximal

$S = \{ v_3, v_7, v_8 \}$

Step 5
Schedule vertices in S to time step l

Set vertices in S to start at 2

Step 6
l = l + 1

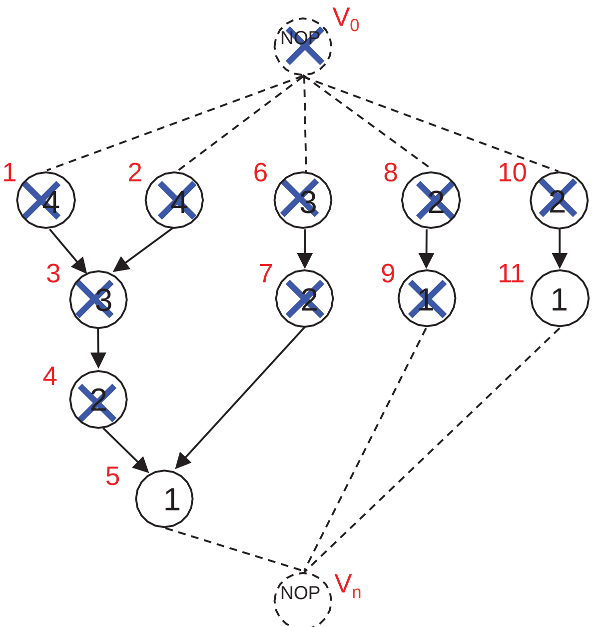
l = 2 + 1 = 3

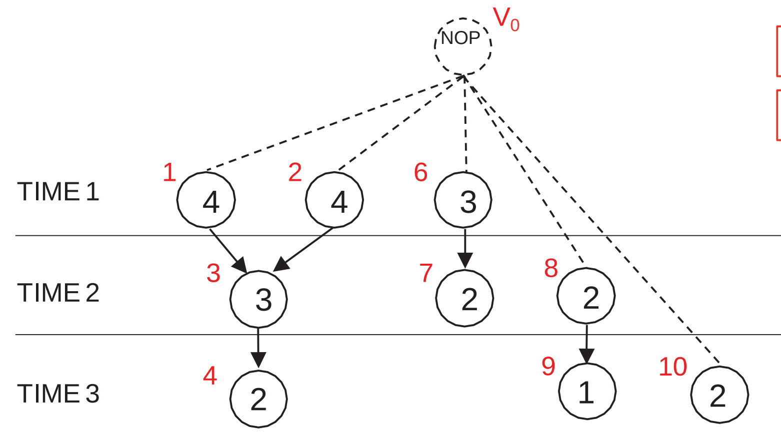
Step 7
Have all vertices been scheduled yet?

No. Repeat loop.

Hu's Algorithm

Example 1





a = 3

l = 3

Step 3
U = unscheduled vertices in V without predecessors or whose predecessors have been scheduled

$U = \{ v_4, v_9, v_{10} \}$

Step 4
S = subset set of vertices in U, no more than a, where labels are maximal

$S = \{ v_4, v_9, v_{10} \}$

Step 5
Schedule vertices in S to time step l

Set vertices in S to start at 3

Step 6
l = l + 1

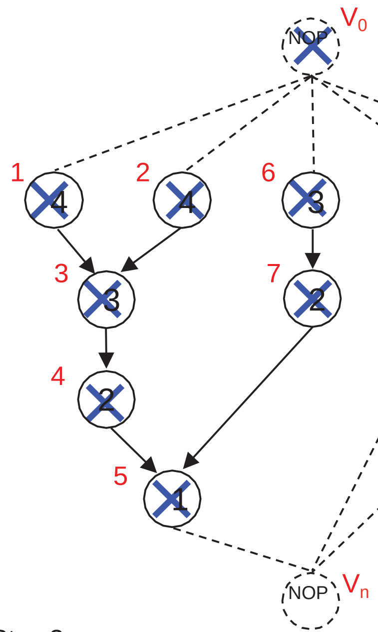
l = 3 + 1 = 4

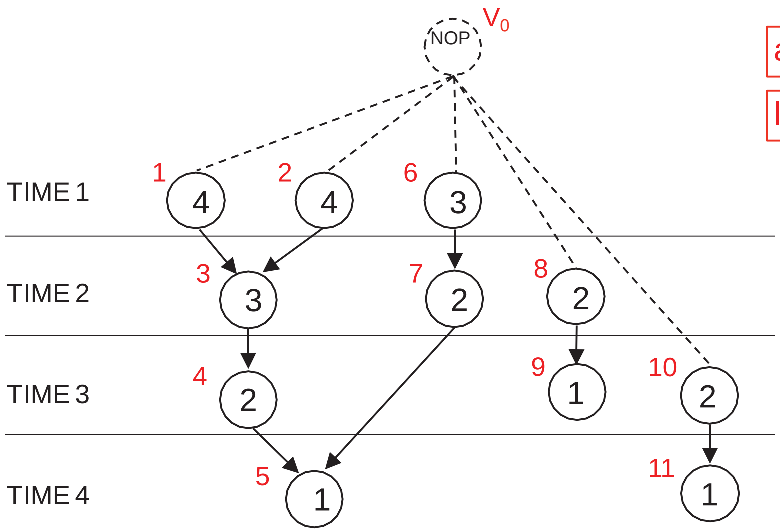
Step 7
Have all vertices been scheduled yet?

No. Repeat loop.

Hu's Algorithm

Example 1





a = 3

l = 4

Step 3

U = unscheduled vertices in V without predecessors or whose predecessors have been scheduled

U = { v_5, v_{11} }

Step 4

S = subset set of vertices in U, no more than a, where labels are maximal

S = { v_5, v_{11} }

Step 5

Schedule vertices in S to time step l

Set vertices in S to start at 4

Step 6

l = l + 1

l = 4 + 1 = 5

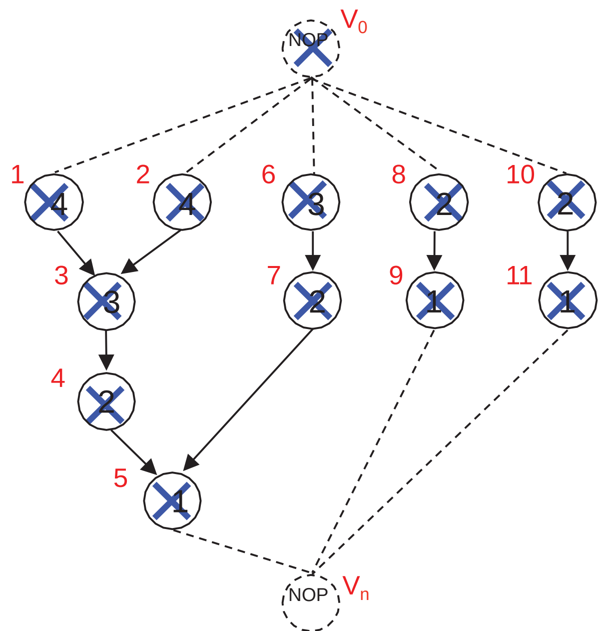
Step 7

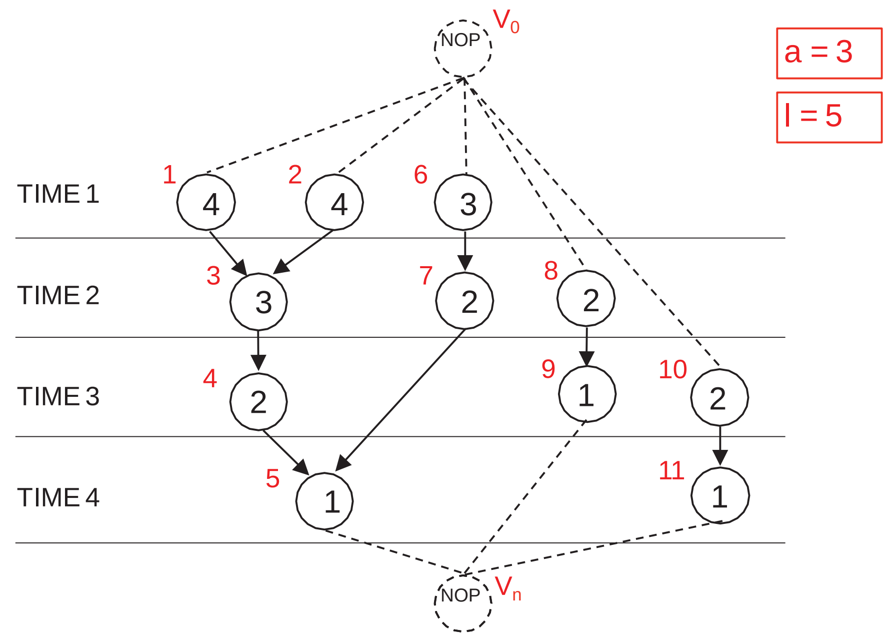
Have all vertices been scheduled yet?

No. Repeat loop.

Hu's Algorithm

Example 1





Step 3
U = unscheduled vertices in V without predecessors or whose predecessors have been scheduled

$U = \{ v_N \}$

Step 4
S = subset set of vertices in U, no more than a, where labels are maximal

$S = \{ v_N \}$

Step 5
Schedule vertices in S to time step l

Set vertices in S to start at 5

Step 6
l = l + 1

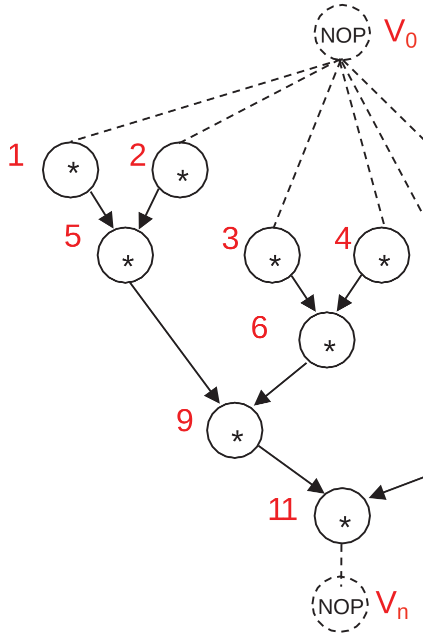
l = 5 + 1 = 6

Step 7
Have all vertices been scheduled yet?

Yes. We are done!

Hu's Algorithm

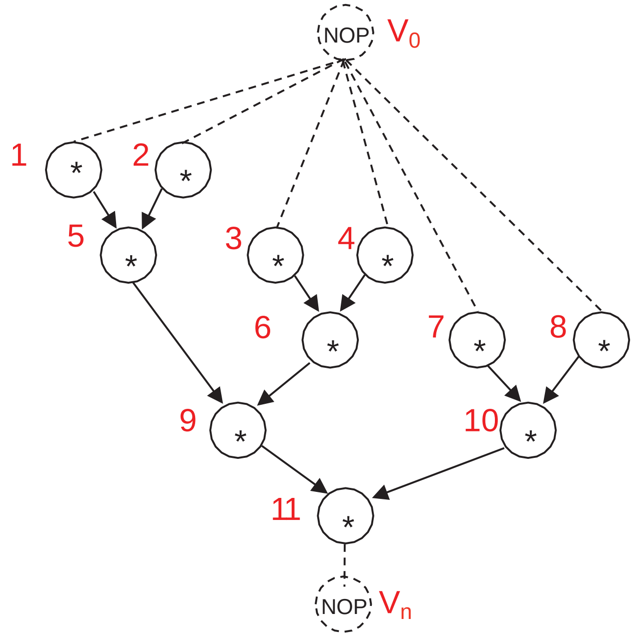
Example 2



a = 4

Hu's Algorithm

Example 3



a = 2

Additional Scheduling Considerations

- Hu's algorithm
  - Assumes one resource handles all possible operations
  - Assumes all operations have 1 unit delay

- Most scheduling problems have additional considerations
  - What happens when we have more than one type of task/operation?
  - What happens when a task/operation takes more than 1 unit delay?

- Increased problem space, difficult problem to solve efficiently
  - Many heuristics have been developed to address these problems
    - Minimum-latency, resource-constrained scheduling
    - Minimum-resource, latency-constrained scheduling

We consider one such heuristic from a family of heuristics called *list scheduling* that looks at the minimum-latency, resource-constrained scheduling problem